Layout Randomization and Nondeterminism

Layout Randomization and Nondeterminism

Martín Abadi[1], Jérémy Planul[2], and Gordon D. Plotkin[1,3]

[1] Microsoft Research
[2] Department of Computer Science, Stanford University
[3] LFCS, School of Informatics, University of Edinburgh

Abstract. In security, layout randomization is a popular, effective attack mitigation technique. Recent work has aimed to explain it rigorously, focusing on deterministic systems. In this paper, we study layout randomization in the presence of nondeterministic choice. We develop a semantic approach based on denotational models and the induced notions of contextual public observation, characterized by simulation relations. This approach abstracts from language details, and helps manage the delicate interaction between nondeterminism and probability. In particular, memory access probabilities are not independent, but rather depend on a fixed probability distribution over memory layouts; we therefore model probability using random variables rather than any notion of probabilistic powerdomain.

1 Introduction

Randomization has important applications in security, ranging from probabilistic cryptographic schemes [1] to the introduction of artificial diversity in low-level software protection [2]. Developing rigorous models and analyses of the systems that employ randomization can be challenging, not only because of the intrinsic difficulty of reasoning about probabilities but also because these systems typically exhibit many other interesting features. Some of these features, such as assumed bounds on the capabilities and the computational complexity of attackers, stem directly from security considerations. Others, such as nondeterminism, need not be specifically related to security, but arise because of the generality of the ambient computational models, which may for example include nondeterministic scheduling for concurrent programs and for network protocols.

The form of randomization that we explore in this paper is layout randomization in software systems (e.g., [3–5]). Layout randomization refers to a body of widely used techniques that place data and code randomly in memory. In practice, these techniques effectively thwart many attacks that assume knowledge of the location of data and code. Recent research by the authors and others aims to develop rigorous models and proofs

for layout randomization [6–9]. The research to date has focused on deterministic, sequential programs. Here, we consider layout randomization for programs that may make nondeterministic choices.

We phrase our study in terms of a high-level language in which variables are abstract (symbolic) locations, and a low-level language in which they are mapped to random natural-number addresses in memory. Both languages include a standard construct for nondeterministic choice. We give models for the languages. For each language, we also define a contextual implementation relation. Intuitively, a context may represent an attacker, so contextual implementation relations may serve, in particular, for expressing standard security properties. We characterize contextual implementation relations in terms of semantic simulation relations (so-called logical relations). Throughout, the low-level relations are probabilistic. Via the simulation relations, we obtain a semantic correspondence between the high-level and low-level worlds. Basically, simulation relations in one world induce simulation relations in the other, and therefore contextual implementation in one world implies contextual implementation in the other.

Thus, our approach emphasizes semantic constructions. In comparison with prior syntactic work, arguments via models arguably lead to more satisfying security arguments, independent of superficial details of particular languages (as layout randomization is largely language-agnostic in practice). They also help reconcile probabilities and nondeterminism, which have a rich but thorny interaction.

Some of the difficulties of this interaction have been noticed in the past. For instance, in their development of a framework for the analysis of security protocols [10, Section 2.7], Lincoln et al. observed:

> our intention is to design a language of communicating processes so that an adversary expressed by a set of processes is restricted to probabilistic polynomial time. However, if we interpret parallel composition in the standard nondeterministic fashion, then a pair of processes may nondeterministically "guess" any secret information.

They concluded:

> Therefore, although nondeterminism is a useful modeling assumption in studying correctness of concurrent programs, it does not seem helpful for analyzing cryptographic protocols.

Thus, they adopted a form of probabilistic scheduling, and excluded nondeterminism. In further work, Mitchell et al. [11] refined the framework,

in particular defining protocol executions by reference to any polynomial-time probabilistic scheduler that operates uniformly over certain kinds of choices. The uniformity prevents collusion between the scheduler and an attacker. Similarly, Canetti et al. [12] resolved nondeterminism by task schedulers, which do not depend on dynamic information generated during probabilistic executions; they thus generated sets of trace distributions, one for each task schedule.

From a semantic perspective, a nondeterministic program denotes a function that produces a set of possible outcomes; equally, a probabilistic program represents a function that produces a distribution over outcomes. Rigorous versions of these statements can be cast in terms of powerdomains and probabilistic powerdomains [13]. In principle, a nondeterministic and probabilistic program may represent either a function producing a set of distributions over outcomes or else one producing a distribution over sets of outcomes. However it seems that only the former option, where nondeterministic choice is resolved before probabilistic choice, leads to a satisfactory theory if, for example, one wishes to retain all the usual laws for both forms of nondeterminism [14–16].

To illustrate these options, imagine a two-player game in which Player I chooses a bit b_{I} at random, Player II chooses a bit b_{II} nondeterministically, and Player I wins if and only if $b_{\mathrm{I}} = b_{\mathrm{II}}$. The system composed of the two players may be seen as producing a set of distributions or a distribution on sets of outcomes.

- With the former view, we can say that, in each possible distribution, Player I wins with probability 1/2.
- On the other hand, with the latter view, we can say only that, with probability 1, Player I may win and may lose.

The former view is preferable in a variety of security applications, in which we may wish to say that no matter what an attacker does, or how nondeterministic choices are resolved, some expected property holds with high probability.

However, in our work, it does not suffice to resolve nondeterministic choice before probabilistic choice, as we explain in detail below, fundamentally because the probabilistic choices that we treat need not be independent. Instead, we construct a more sophisticated model that employs random variables, here maps from memory layouts to outcomes. The memory layouts form the sample space of the random variables, and, as usual, one works relative to a given distribution over the sample space.

Beyond the study of layout randomization, it seems plausible that an approach analogous to ours could be helpful elsewhere in security analysis.

Our models may also be of interest on general grounds, as a contribution to a long line of research on programming-language semantics for languages with nondeterministic and probabilistic choice. Specifically, the models support a treatment of dependent probabilistic choice combined with nondeterminism, which as far as we know has not been addressed in the literature. Finally, the treatment of contextual implementation relations and simulation relations belongs in a long line of research on refinement.

This paper is a full version of a conference paper [17] of the same title. The main differences are that proofs are presented in full (except in some routine or evident cases) and that an inconsistency between the operational and denotational semantics of the low-level language has been corrected by an alteration to its operational semantics.

Contents

In Section 2 we review some preliminary material on cpos.

In Section 3, we consider a high-level language, with abstract locations, standard imperative constructs, and nondeterminism, and describe its denotational and operational semantics. We define a contextual implementation relation with respect to contexts that represent attackers, which we call public contexts; for this purpose, we distinguish public locations, which attackers can access directly, from private locations. We also define a simulation relation, and prove that it coincides with the contextual implementation relation. The main appeal of the simulation relation, as usual, is that it does not require reasoning about all possible contexts.

In Section 4, we similarly develop a lower-level language in which programs may use natural-number memory addresses (rather than abstract locations). Again, we define a denotational semantics, an operational semantics, a contextual implementation relation, and a simulation relation. These definitions are considerably more delicate than those of the high-level language, in particular because they refer to layouts, which map abstract locations to concrete natural-number addresses, and which may be chosen randomly (so we often make probabilistic statements).

In Section 5, we relate the high-level and the low-level languages. We define a simple compilation function that maps from the former to the latter. We then establish that if two high-level commands are in the contextual implementation relation, then their low-level counterparts are also in the contextual implementation relation. The proof leverages simulation relations. In semantics parlance, this result is a full-abstraction theorem;

the use of public contexts that represent attackers, however, is motivated by security considerations, and enable us to interpret this theorem as providing a formal security guarantee for the compilation function, modulo a suitable random choice of memory layouts.

Finally, in Section 6 we conclude by discussing some related and further work.

2 Preliminaries on Cpos

We take a cpo to be a partial order P closed under increasing ω-sups, and consider sets to be cpos with the discrete ordering. We write $P_\perp$ for the lift of P, viz. P extended by the addition of a least element, $\perp$. Products $P \times Q$ and function spaces $P \to Q$ (which we may also write as Q^P) are defined as usual, with the function space consisting of all continuous functions (those monotonic functions preserving the ω-lubs).

We use the lower, or Hoare, powerdomain $\mathcal{H}(P)$, which consists of the nonempty, downwards, and ω-sup-closed subsets of P, ordered by inclusion. The lower powerdomain is the simplest of the three powerdomains, and models "may" or "angelic" nondeterminism; the others (upper and convex) may also be worth investigating.

For any nonempty subset X of P, we write $X\downarrow$ for the downwards closure $\{y \mid \exists x \in X.\, y \leq x\}$ of X. We also write X^* for the downwards and ω-sup closure of X (which is typically the same as $X\downarrow$ in the instances that arise below).

Both $\mathcal{H}(-)$ and $\mathcal{H}(-_\perp)$ are monads (those for lower nondeterminism, and lower nondeterminism and nontermination, respectively). The unit of the former is $x \mapsto \{x\}\downarrow$ and continuous maps $f : P \to \mathcal{H}(Q)$ have extensions $f^\dagger : \mathcal{H}(P) \to \mathcal{H}(Q)$ given by:

$$f^\dagger(X) = (\bigcup_{x \in X} f(x))^*$$

The unit of the latter is $x \mapsto \{x\}\downarrow$ and continuous maps $f : P \to \mathcal{H}(Q_\perp)$ have extensions $f^\dagger : \mathcal{H}(P_\perp) \to \mathcal{H}(Q_\perp)$ given by:

$$f^\dagger(X) = \{\perp\} \cup (\bigcup_{x \in X \setminus \{\perp\}} f(x))^*$$

3 The High-Level Language

In this section, we define our high-level language. In this language, locations are symbolic names, and we use an abstract store to link those locations to their contents, which are natural numbers.

For simplicity, the language lacks data structures and higher-order features. Therefore, locations cannot contain arrays or functions (cf. [9]), except perhaps through encodings. So the language does not provide a direct model of overflows and code-injection attacks, for instance.

There are many other respects in which our languages and their semantics are not maximally expressive, realistic, and complex. They are however convenient for our study of nondeterminism and of the semantic approach to layout randomization.

3.1 Syntax and Informal Semantics

The syntax of the high-level language includes categories for natural-number expressions, boolean expressions, and commands:

$$
\begin{array}{rcl}
e & ::= & k \mid !l_{\mathtt{loc}} \mid e + e \mid e * e \\
b & ::= & e \leq e \mid \neg b \mid \mathrm{tt} \mid \mathrm{ff} \mid b \vee b \mid b \wedge b \\
c & ::= & l_{\mathtt{loc}} := e \mid \mathtt{if}\, b\, \mathtt{then}\, c\, \mathtt{else}\, c \mid \mathtt{skip} \mid c; c \mid c + c \mid \mathtt{while}\, b\, \mathtt{do}\, c
\end{array}
$$

where k ranges over numerals, and l over a given finite set of store locations Loc. Natural-number expressions are numerals, dereferencing of memory locations, sums, or products. Boolean expressions are inequalities on natural-number expressions, negations, booleans, disjunctions, or conjunctions. Commands are assignments at a location, conditionals, skip, sequences, nondeterministic choices, or while loops. Command contexts $C[\,]$ are commands with holes; we write $C[c]$ for the command obtained by filling all the holes in $C[\,]$ with c. We further use trivial extensions of this language, in particular with additional boolean and arithmetic expressions.

We assume that the set of store locations Loc is the union of two disjoint sets of locations PubLoc (public locations) and PriLoc (private locations). Let c be a command or a command context. We say that c is public if it does not contain any occurrence of $l_{\mathtt{loc}} := v$ or $!l_{\mathtt{loc}}$ for $l \in \mathrm{PriLoc}$. As in previous work [7], we model attackers by such public commands and command contexts; thus, attackers have direct access to public locations but not, by default, to private locations.

The distinction between public and private locations is directly analogous to that between external and internal state components in automata and other specification formalisms (e.g., [18]). It also resembles distinctions in information-flow systems, which often categorize variables into levels (e.g., [19]), and typically aim to prevent flows of information from

"high" to "low" levels. We do not impose any such information-flow constraint: we permit arbitrary patterns of use of public and private locations. Nevertheless, we sometimes use h for a private location and l for a public location, and also associate the symbols H and L with private and public locations, respectively.

3.2 Denotational Semantics

A store s is a function from the finite set Loc of store locations to natural numbers. When Loc consists solely of h and l, for example, we write $(h \mapsto m, l \mapsto n)$ for the store that maps h to m and l to n. A public (private) store is a function from PubLoc (PriLoc) to natural numbers. We write S for the set of stores, S_L for the set of public stores, and S_H for the set of private stores. The following natural functions restrict the store to its public and private locations:

$$S_L \xleftarrow{L} S \xrightarrow{H} S_H$$

We write s_L for $L(s)$ and $s =_L s'$ when $s_L = s'_L$, and similarly for H.

The denotational semantics

$$[\![e]\!] : \mathrm{Store} \to \mathbb{N} \qquad [\![b]\!] : \mathrm{Store} \to \mathbb{B}$$

of expressions are defined as usual with, in particular, $[\![!l_{\mathtt{loc}}]\!](s) = s(l)$. The denotational semantics

$$[\![c]\!] : S \to \mathcal{H}(S_\perp)$$

of commands is given in Figure 1, where the semantics of the while loop is the standard least-fixed point one.

$$\begin{array}{lll}
[\![l_{\mathtt{loc}} := e]\!](s) & = \eta(s[l \mapsto [\![e]\!](s)]) & [\![\mathsf{skip}]\!](s) = \eta(s) \\
[\![\mathtt{if}\, b\, \mathtt{then}\, c\, \mathtt{else}\, c']\!](s) & = \begin{cases} [\![c]\!](s) & ([\![b]\!](s) = \mathrm{tt}) \\ [\![c']\!](s) & ([\![b]\!](s) = \mathrm{ff}) \end{cases} & \begin{array}{l} [\![c; c']\!](s) = [\![c']\!]^\dagger([\![c]\!](s)) \\ [\![c + c']\!](s) = [\![c]\!](s) \cup [\![c']\!](s) \end{array} \\
[\![\mathtt{while}\, b\, \mathtt{do}\, c]\!] & = \mu\, \theta : S \to \mathcal{H}(S_\perp).\, \lambda s : S.\, \begin{cases} \eta(s) & ([\![b]\!](s) = \mathrm{ff}) \\ \theta^\dagger([\![c]\!](s)) & ([\![b]\!](s) = \mathrm{tt}) \end{cases} &
\end{array}$$

Fig. 1. High-level denotational semantics

Example 1. Consider the two commands:

$$c_0 = (h := 1; l := \neg !l) + (h := 0) \qquad c_1 = (h := 1; l := 1) + (h := 0; l := 0)$$

According to the semantics, $[\![c_0]\!]$ maps any store mapping l to 1 to the set $\{(h \mapsto 1, l \mapsto 0), (h \mapsto 0, l \mapsto 1)\}\downarrow$, and any store where l is 0 to the set $\{(h \mapsto 1, l \mapsto 1), (h \mapsto 0, l \mapsto 0)\}\downarrow$, while $[\![c_1]\!]$ maps any store to the set $\{(h \mapsto 1, l \mapsto 1), (h \mapsto 0, l \mapsto 0)\}\downarrow$. In sum, we may write:

$$\begin{aligned}
[\![c_0]\!](h \mapsto _, l \mapsto 1) &= \{(h \mapsto 1, l \mapsto 0), (h \mapsto 0, l \mapsto 1)\}\downarrow \\
[\![c_0]\!](h \mapsto _, l \mapsto 0) &= \{(h \mapsto 1, l \mapsto 1), (h \mapsto 0, l \mapsto 0)\}\downarrow \\
[\![c_1]\!](h \mapsto _, l \mapsto _) &= \{(h \mapsto 1, l \mapsto 1), (h \mapsto 0, l \mapsto 0)\}\downarrow
\end{aligned}$$

Since the two commands act differently on stores, they do not have the same semantics. However, when one observes only public locations, the apparent behavior of both commands is the same: they nondeterministically write 0 or 1 to l. This similarity will be made rigorous in Example 2. □

3.3 Operational Semantics

The high-level language has a straightforward big-step operational semantics. In this semantics, a high-level state is a pair $\langle c, s\rangle$ of a command and a store or, marking termination, just a store s. The transition relation is a binary relation $\langle c, s\rangle \Rightarrow s$ between such states. Figure 2 gives the rules for $\Rightarrow$. (Note that we treat expressions denotationally; as we wish to focus on commands, this treatment avoids some extra complexity.)

$$\langle l_{\mathtt{loc}} := e, s\rangle \Rightarrow s[l \mapsto [\![e]\!]_s] \qquad \frac{[\![b]\!]_s = \mathrm{tt} \quad \langle c, s\rangle \Rightarrow s'}{\langle \mathtt{if}\, b\, \mathtt{then}\, c\, \mathtt{else}\, c', s\rangle \Rightarrow s'}$$

$$\frac{[\![b]\!]_s = \mathrm{ff} \quad \langle c', s\rangle \Rightarrow s'}{\langle \mathtt{if}\, b\, \mathtt{then}\, c\, \mathtt{else}\, c', s\rangle \Rightarrow s'} \qquad \langle \mathsf{skip}, s\rangle \Rightarrow s \qquad \frac{\langle c, s\rangle \Rightarrow s' \quad \langle c', s'\rangle \Rightarrow s''}{\langle c; c', s\rangle \Rightarrow s''}$$

$$\frac{\langle c, s\rangle \Rightarrow s'}{\langle c + c', s\rangle \Rightarrow s'} \qquad \frac{\langle c', s\rangle \Rightarrow s'}{\langle c + c', s\rangle \Rightarrow s'}$$

$$\frac{[\![b]\!]_s = \mathrm{ff}}{\langle \mathtt{while}\, b\, \mathtt{do}\, c, s\rangle \Rightarrow s} \qquad \frac{[\![b]\!]_s = \mathrm{tt} \quad \langle c, s\rangle \Rightarrow s' \quad \langle \mathtt{while}\, b\, \mathtt{do}\, c, s'\rangle \Rightarrow s''}{\langle \mathtt{while}\, b\, \mathtt{do}\, c, s\rangle \Rightarrow s''}$$

Fig. 2. High-level operational semantics

The following proposition links the operational and denotational semantics of the high-level language.

Proposition 1 (High-level operational/denotational consistency). *For any high-level command c and store s, we have:*

$$[\![c]\!](s) = \{s' | \langle c, s\rangle \Rightarrow s'\} \cup \{\bot\}$$

Proof. In one direction, using rule-induction, one shows that if $\langle c, s\rangle \Rightarrow s'$ then $s' \in [\![c]\!](s)$. In the other direction one shows, by structural induction on loop-free commands, that if $s' \in [\![c]\!](s)$ then $\langle c, s\rangle \Rightarrow s'$. One then establishes the result for all commands, including while loops, by considering their iterates, where loops are unwound a finite number of times. We omit details.

3.4 Implementation Relations and Equivalences

We next define the contextual pre-order that arises from the notion of public observation. We then give an equivalent simulation relation, with which it is easier to work as it does not refer to contexts.

Contextual Pre-order. We introduce a contextual pre-order $\sqsubseteq_L$ on commands. Intuitively, $c \sqsubseteq_L c'$ may be interpreted as saying that c "refines" (or "implements") c', in the sense that the publicly observable outcomes that c can produce are a subset of those that c' permits, in every public context and from every initial store. Thus, let $f = [\![C[c]]\!]$ and $f' = [\![C[c']]\!]$ for an arbitrary public context C, and let s_0 be a store; then for every store s in $f(s_0)$ there is a store s' in $f'(s_0)$ that coincides with s on public locations. Note that we both restrict attention to public contexts and compare s and s' only on public locations.

We define $\sqsubseteq_L$ and some auxiliary relations as follows:

- For $X \in \mathcal{H}(S_\bot)$, we set:

$$X_L = \{s_L \mid s \in X \setminus \{\bot\}\} \cup \{\bot\}$$

- For $f, f' : S \to \mathcal{H}(S_\bot)$, we write that $f \leq_L f'$ when, for every store s_0, we have $f(s_0)_L \leq f'(s_0)_L$.
- Let c and c' be two commands. We write that $c \sqsubseteq_L c'$ when, for every public command context C, we have $[\![C[c]]\!] \leq_L [\![C[c']]\!]$.

Straightforwardly, this contextual pre-order relation yields a notion of contextual equivalence with respect to public contexts.

Simulation. We next give the simulation relation $\preceq$. As in much previous work, one might expect the simulation relation between two commands c and c' to be a relation on stores that respects the observable parts of these stores, and such that if s_0 is related to s_1 and c can go from s_0 to s_0' then there exists s_1' such that s_0' is related to s_1' and c' can go from s_1 to s_1'. In our setting, respecting the observable parts of stores means that related stores give the same values to public locations (much like refinement mappings preserve externally visible state components [18], and low-bisimulations require equivalence on low-security variables [19]).

Although this idea could lead to a sound proof technique for the contextual pre-order, it does not suffice for completeness. Indeed, forward simulations, of the kind just described, are typically incomplete on their own for nondeterministic systems. They can be complemented with techniques such as backward simulation, or generalized (e.g., [18, 20, 21]).

Here we develop one such generalization. Specifically, we use relations on sets of stores. We build them from relations over $\mathcal{H}(S_{H\perp})$ as a way of ensuring the condition that public locations have the same values, mentioned above. We also require other standard closure conditions. Our relations are similar to the ND measures of Klarlund and Schneider [20]. Their work takes place in an automata-theoretic setting; automata consist of states (which, intuitively, are private) and of transitions between those states, labeled by events (which, intuitively, are public). ND measures are mappings from states to sets of finite sets of states, so can be seen as relations between states and finite sets of states. The finiteness requirement, which we do not need, allows a fine-grained treatment of infinite execution paths via König's Lemma.

First, we extend relations R over $\mathcal{H}(S_{H\perp})$ to relations R^+ over $\mathcal{H}(S_\perp)$, as follows. For any $X \in \mathcal{H}(S_\perp)$ and $s \in S_L$, we define $X_s \in \mathcal{H}(S_{H\perp})$ by:

$$X_s = \{s'_H \mid s' \in X, s'_L = s\} \cup \{\perp\}$$

and then we define R^+ by:

$$X R^+ Y \equiv_{\text{def}} \forall s \in S_L.\, (X_s \neq \{\perp\} \Rightarrow Y_s \neq \{\perp\}) \wedge X_s R Y_s$$

If R is reflexive (respectively, is closed under increasing ω-sups; is right-closed under $\leq$; is closed under binary unions) the same holds for R^+. Also, if $X R^+ Y$ then $X_L \leq Y_L$.

For any $f, f' : S_\perp \to \mathcal{H}(S_\perp)$ and relation R over $\mathcal{H}(S_{H\perp})$ we write that $f \preceq_R f'$ when:

$$\forall X, Y \in \mathcal{H}(S_\perp).\, X R^+ Y \Rightarrow f^\dagger(X) R^+ f'^\dagger(Y)$$

If $f \preceq_R f'$ holds then we have $f \leq_L f'$ (as follows from the fact that $X_L \leq Y_L$ holds if XR^+Y does).

Finally, we write that $f \preceq f'$ if $f \preceq_R f'$ for some reflexive R closed under increasing ω-sups, right-closed under $\leq$, and closed under binary unions.

Contextual Pre-order vs. Simulation. The contextual pre-order coincides with the simulation relation, as we now show. We break the proof into two parts.

Lemma 1. *Let c and c' be two commands of the high-level language such that $[\![c]\!] \preceq [\![c']\!]$. Then $c \sqsubseteq_L c'$ holds.*

Proof. Let c_0 and c_1 be two commands such that $[\![c_0]\!] \preceq_R [\![c_1]\!]$, with R a reflexive relation over $\mathcal{H}(S_{H\perp})$ closed under increasing ω-sups, right-closed under $\leq$, and closed under binary unions, and let C be a public command context. We prove that $[\![C[c_0]]\!] \preceq_R [\![C[c_1]]\!]$ by induction on the size of C, considering the possible forms of C:

1. $l_{\mathtt{loc}} := e$: Suppose that XR^+Y. As e is public, for every s, $[\![e]\!](s)$ only depends on s_L. As l is also public, $[\![l_{\mathtt{loc}} := e]\!](s) = \{s'\} \cup \{\bot\}$, for every s, where $s' =_H s$ and s'_L depends only on s_L. Therefore, for every $s_0 \in S_L$, let $S_0 \subseteq S$ be the (possibly empty) set

$$\{s \in S \mid \exists s' \in S, [\![l_{\mathtt{loc}} := e]\!](s) = \{s'\} \cup \{\bot\} \text{ and } s'_L = s_0\}$$

   We then have that $[\![l_{\mathtt{loc}}{:=}e]\!](X)_{s_0} = (\bigcup_{s \in S_0} X_{s_L}) \cup \{\bot\}$ and also that $[\![l_{\mathtt{loc}}{:=}e]\!](Y)_{s_0} = (\bigcup_{s \in S_0} Y_{s_L}) \cup \{\bot\}$.
   To see that $[\![l_{\mathtt{loc}}{:=}e]\!](X) R^+ [\![l_{\mathtt{loc}}{:=}e]\!](Y)$, choose $s_0 \in S_L$, and define S_0 as above. Then, if $[\![l_{\mathtt{loc}}{:=}e]\!](X) \neq \{\bot\}$ there is an $s \in S_0$ such that $X_{s_L} \neq \{\bot\}$. But then, as XR^+Y, we have that $Y_{s_L} \neq \{\bot\}$, and so $[\![l_{\mathtt{loc}}{:=}e]\!](Y) \neq \{\bot\}$.
   Finally we have to check that $[\![l_{\mathtt{loc}}{:=}e]\!](X)_{s_0} R [\![l_{\mathtt{loc}}{:=}e]\!](Y)_{s_0}$. That follows from the above two formulas, as XR^+Y and R is closed under countable unions and reflexive.
2. $\mathtt{if}\, b\, \mathtt{then}\, C_{\mathtt{tt}}\, \mathtt{else}\, C_{\mathtt{ff}}$: Suppose that XR^+Y. Define $X_{\mathtt{tt}} \subseteq X$ to be the set $\{s \in X | [\![b]\!](s) = \mathtt{tt}\} \cup \{\bot\}$ and define $Y_{\mathtt{tt}}$, $X_{\mathtt{ff}}$, and $Y_{\mathtt{ff}}$ similarly. As b is public, for every s, $[\![b]\!](s)$ only depends on s_L, and so, for any $s_0 \in S_L$ we have:

$$(X_{\mathtt{tt}})_{s_0} = \begin{cases} X_{s_0} \cup \{\bot\} & (\exists s \in X_{\mathtt{tt}}.\, s_0 = s_L) \\ \{\bot\} & (\text{otherwise}) \end{cases}$$

and similar equations hold for Y_{tt}, X_{ff}, and Y_{ff}. We then check that $X_{\mathrm{tt}} R^+ Y_{\mathrm{tt}}$ and $X_{\mathrm{ff}} R^+ Y_{\mathrm{ff}}$ much as in the previous case.
We have

$$[\![\mathtt{if}\, b\, \mathtt{then}\, C_{\mathrm{tt}}[c_0]\, \mathtt{else}\, C_{\mathrm{ff}}[c_0]]\!]^\dagger(X) = [\![C_{\mathrm{tt}}[c_0]]\!]^\dagger(X_{\mathrm{tt}}) \cup [\![C_{\mathrm{ff}}[c_0]]\!]^\dagger(X_{\mathrm{ff}})$$

and similarly for Y. By induction, $[\![C_{\mathrm{tt}}[c_0]]\!]^\dagger(X_{\mathrm{tt}}) R^+ [\![C_{\mathrm{tt}}[c_1]]\!]^\dagger(Y_{\mathrm{tt}})$ and similarly for ff. As R^+ is closed under binary unions, we conclude.

3. skip: The conclusion is immediate as $[\![\mathsf{skip}]\!]^\dagger$ is the identity.
4. $C'; C''$: Here we have:

$$[\![C'[c_0]; C''[c_0]]\!]^\dagger = ([\![C''[c_0]]\!]^\dagger [\![C'[c_0]]\!])^\dagger = [\![C''[c_0]]\!]^\dagger [\![C'[c_0]]\!]^\dagger$$

and the same holds for c_1, and so the conclusion follows using the induction hypothesis.
5. $C' + C''$: Here, as R^+ is closed under binary unions, the conclusion follows using the induction hypothesis.
6. $\mathtt{while}\, b\, \mathtt{do}\, C_w$: Define iterates $C^{(n)}$ by setting:

$$C^{(0)} = \Omega \qquad C^{(n+1)} = \mathtt{if}\, b\, \mathtt{then}\, \mathsf{skip}\, \mathtt{else}\, C_w; C^{(n)}$$

where Ω is some command denoting $\bot$. By induction on n, we have $C^{(n)}[c_0] \preceq_R C^{(n)}[c_1]$: the case $n = 0$ follows as we have $\{\bot\} R^+ \{\bot\}$, and the induction step follows using the same reasoning as in the second, third, and fourth cases of the proof.
But then, as we have

$$C[c_0] = \bigvee_{n \geq 0} C^{(n)}[c_0]$$

and the same holds for c_1, the conclusion follows using the fact that R^+ is closed under increasing ω-sups.
7. [ ]: We have $C[c_0] = c_0$ and $C[c_1] = c_1$, and the conclusion follows using the hypothesis.

This concludes the proof since it follows from $[\![C[c_0]]\!] \preceq_R [\![C[c_1]]\!]$ that $[\![C[c_0]]\!] \leq_L [\![C[c_1]]\!]$.

We need a lemma in order to prove the converse of Lemma 1.

Lemma 2. *Let R_i $(i \geq 0)$ be relations on $\mathcal{H}(S_{H\bot})$ such that if $X R_i Y$ holds then $X \neq \{\bot\}$ implies $Y \neq \{\bot\}$. Let R be the closure of the union of the R_i under increasing ω-sups, binary union, and right-closure under $\leq$. Then R^+ is the closure of the union of the relations R_i^+ under increasing ω-sups, binary union, and right-closure under $\leq$.*

Proof. As $-^+$ is evidently monotone, R^+ contains the R_i^+. Next, we know that if a relation S on $\mathcal{H}(S_{H\perp})$ is closed under any one of increasing ω-sups, binary unions, or right-closure under $\leq$, then so is S^+. So R^+ is closed under increasing ω-sups and binary unions, and right-closed under $\leq$. It is therefore included in the closure of the union of the R_i^+ under increasing ω-sups, binary unions, and right-closure under $\leq$.

For the converse, suppose that UR^+W to show that U and W are related in the closure of the union of the R_i^+ under increasing ω-sups, binary unions, and right-closure under $\leq$. For any given s in S_L, by definition of $-^+$, $U_s R W_s$, and so, by the definition of R, there is a set $J^{(s)} \subseteq \mathbb{N}$, and relations $X_j^{(s)} R_{i_j^{(s)}} Y_j^{(s)}$ such that $U_s = \bigcup_{j \in J^{(s)}} X_j^{(s)}$ and $W_s \supseteq \bigcup_{j \in J^{(s)}} Y_j^{(s)}$. We may assume without loss of generality that the $J^{(s)}$ are disjoint.

Let

$$\begin{aligned} J &= \textstyle\bigcup_{s \in S_L} J^{(s)} \\ X_j &= \{s' \mid s'_H \in X_j^{(s)}, s'_L = s\} \cup \{\perp\} && (j \in J^{(s)}) \\ Y_j &= \{s' \mid s'_H \in Y_j^{(s)}, s'_L = s\} \cup \{\perp\} && (j \in J^{(s)}) \\ i_j &= i_j^{(s)} && (j \in J^{(s)}) \end{aligned}$$

We verify that $(X_j)_s$ is equal to $X_j^{(s)}$, if j is in $J^{(s)}$, and equal to $\{\perp\}$, otherwise, and similarly for the $(Y_j)_s$. Consequently, $U = \bigcup_j X_j$, $W \supseteq \bigcup_j Y_j$, and for all s in S_L, $(X_j)_s R_{i_j} (Y_j)_s$. Since, by hypothesis, if $(X_j)_s R_{i_j} (Y_j)_s$ holds then $(X_j)_s \neq \{\perp\}$ implies $(Y_j)_s \neq \{\perp\}$, we note that $X_j R_{i_j}^+ Y_j$. We conclude that U and W are related as required.

We also need some notation. Assume a fixed enumeration $x_1 \ldots x_n$ of PubLoc. Then, given high-level commands c_i $(i = 1, \ldots, n)$ we write $[c_x \mid x \in \text{PubLoc}]$ for the high-level command $c_{x_1}; \ldots; c_{x_n}$. As usual, we abbreviate $\texttt{if}\, b\, \texttt{then}\, c\, \texttt{else skip}$ to $\texttt{if}\, b\, \texttt{then}\, c$. We can now show:

Lemma 3. *Let c and c' be two commands of the high-level language such that $c \sqsubseteq_L c'$. Then $[\![c]\!] \preceq [\![c']\!]$ holds.*

Proof. Let c_0 and c_1 be two commands such that $c_0 \sqsubseteq_L c_1$. We define relations R_i $(i \geq 0)$ on $\mathcal{H}(S_{H\perp})$ as follows:

- for every $X \in \mathcal{H}(S_{H\perp})$, we have $X\ R_0\ X$;
- for every $X, Y \in \mathcal{H}(S_\perp)$, such that $X\ R_i^+\ Y$, and for every $s \in S_L$ we have $[\![c_0]\!]^\dagger(X)_s\ R_{i+1}\ [\![c_1]\!]^\dagger(Y)_s$.

We first prove by induction on i that, if $X\ R_i^+\ Y$, then, for every $s \in X$ such that $s \neq \perp$, there exist a public command context C and $s_0 \in S$ such that $s \in [\![C[c_0]]\!](s_0)$ and $[\![C[c_1]]\!](s_0)_{s_L} \subseteq Y_{s_L}$.

– Suppose that $X\ R_0^+\ Y$. For every s, we let C be skip and s_0 be s. We have $s \in [\![\mathsf{skip}]\!](s)$ and $[\![\mathsf{skip}]\!](s)_{s_L} = \{s, \bot\}_{s_L} \subseteq X_{s_L} = Y_{s_L}$.
– Suppose that $X\ R_{i+1}^+\ Y$. By definition of $X\ R_{i+1}^+\ Y$, and in particular $X_{s_L}\ R_{i+1}\ Y_{s_L}$, there exist $X'\ R_i^+\ Y'$ and $s' \in S_L$ such that $[\![c_0]\!]^\dagger(X')_{s'} = X_{s_L}$ and $[\![c_1]\!]^\dagger(Y')_{s'} = Y_{s_L}$. As s_H in X_{s_L}, by definition of $-^\dagger$, there exist $s'' \in X'$ and $s''' \in S$ such that $s''' \in [\![c_0]\!](s'')$, $s'''_L = s'$ and $s'''_H = s_H$ (note that $s'' \neq \bot$).
By induction on $X'\ R_i\ Y'$ and s'', there exists a public command context C and an $s_0 \in S$ such that both $s'' \in [\![C[c_0]]\!](s_0)$ and $[\![C[c_1]]\!](s_0)_{s''_L} \subseteq Y'_{s''_L}$ hold.
We consider the public command context

$$C' =_{\text{def}} C; [\texttt{if}\ !x_{\texttt{loc}} \neq s''_L(x)\ \texttt{then}\ \Omega \mid x \in \text{PubLoc}]; \\ [\,]; [x := s_L(x) \mid x \in \text{PubLoc}]$$

We have s'' in $[\![C[c_0]]\!](s_0)$, so s''' is in

$$[\![C[c_0]; [\texttt{if}\ !x_{\texttt{loc}} \neq s''_L(x)\ \texttt{then}\ \Omega \mid x \in \text{PubLoc}]; c_0]\!](s_0)$$

so s is in $[\![C'[c_0]]\!](s_0)$.
Also, $[\![C[c_1]]\!](s_0)_{s''_L} \subseteq Y'_{s''_L}$, hence

$$[\![C[c_1]; [\texttt{if}\ !x_{\texttt{loc}} \neq s''_L(x)\ \texttt{then}\ \Omega \mid x \in \text{PubLoc}]]\!](s_0)\ \subseteq\ Y'$$

hence

$$[\![C[c_1]; [\texttt{if}\ !x_{\texttt{loc}} \neq s''_L(x)\ \texttt{then}\ \Omega \mid x \in \text{PubLoc}]; c_1]\!](s_0)_{s'}\ \subseteq\ Y_{s_L}$$

and hence (we rewrite the low variables with their corresponding values in s_L)

$$[\![C'[c_1]]\!](s_0)_{s_L} \subseteq Y_{s_L}$$

We now prove that

$$\forall X, Y \in \mathcal{H}(S_{H\bot}).\, X R_i Y \Rightarrow (X \neq \{\bot\} \Rightarrow Y \neq \{\bot\})$$

For $i = 0$, this follows from the definition of R_0. Otherwise, $i = j+1$, and by definition of (R_i), there exist X', Y', and $s \in S_L$ such that $X' R_j^+ Y'$, $X = [\![c_0]\!]^\dagger(X')_s$, and $Y = [\![c_1]\!]^\dagger(Y')_s$. If $X \neq \{\bot\}$, by definition of $-^\dagger$, there exists $s' \in X'$ such that $[\![c_0]\!](s')_s \neq \{\bot\}$ (note that $s' \neq \bot$). As shown above, since $X' R_j^+ Y'$, there exist a public command context C and $s_0 \in S$ such that $s' \in [\![C[c_0]]\!](s_0)$ and $[\![C[c_1]]\!](s_0)_{s'_L} \subseteq Y'_{s'_L}$.

We let $C' = C[\ ];[\texttt{if}\ !x_{\texttt{loc}} \neq s'_L(x)\ \texttt{then}\ \Omega | x \in \text{PubLoc}];[\ ]$. We have $[\![c_0]\!](s')_s \subseteq [\![C'[c_0]]\!](s_0)_s \neq \{\bot\}$. Also, since C' is a public command context, we have $[\![C'[c_0]]\!](s_0) \leq_L [\![C'[c_1]]\!](s_0)$. Hence $[\![C'[c_1]]\!](s_0)_s \neq \{\bot\}$, and we conclude since $[\![C'[c_1]]\!](s_0)_s \subseteq [\![c_1]\!]^\dagger(Y')_s$.

By definition of R_{i+1}, we have

$$X\ R_i^+\ Y \Rightarrow \forall s \in S_L, [\![c_0]\!]^\dagger(X)_s\ R_{i+1}\ [\![c_1]\!]^\dagger(Y)_s$$

From the result above, we deduce

$$\forall X, Y \in \mathcal{H}(S_\bot).\, X R_i^+ Y \Rightarrow [\![c_0]\!]^\dagger(X) R_{i+1}^+ [\![c_1]\!]^\dagger(Y) \qquad (*)$$

We now let R be the closure of the union of the R_i increasing ω-sups, right-closure under $\leq$ and closure under binary unions. Note that R is reflexive as it contains R_0. By Lemma 2, we then have that R^+ is the closure of the union of the R_i^+ under increasing ω-sups, right-closure under $\leq$, and closure under binary unions. Since every $[\![c]\!]^\dagger$ is monotone and distributes over these unions, and given the property $(*)$ above, we conclude that $[\![c_0]\!] \preceq_R [\![c_1]\!]$.

Lemmas 1 and 3 give us the desired equivalence:

Theorem 1. *Let c and c' be two commands of the high-level language. Then $c \sqsubseteq_L c'$ holds if and only if $[\![c]\!] \preceq [\![c']\!]$ does.*

Example 2. We can verify that c_0 and c_1, introduced in Example 1, are equivalent (with R the full relation). For instance, take S_0 and S_1 to be $\{(h \mapsto 0, l \mapsto 1)\}\downarrow$ and $\{(h \mapsto 1, l \mapsto 1)\}\downarrow$. We have $S_0 R^+ S_1$, and:

$$[\![c_0]\!]^\dagger(S_0) = \{(h \mapsto 1, l \mapsto 0), (h \mapsto 0, l \mapsto 1)\}\downarrow$$
$$[\![c_1]\!]^\dagger(S_1) = \{(h \mapsto 1, l \mapsto 1), (h \mapsto 0, l \mapsto 0)\}\downarrow$$

We can then check that:

$$[\![c_0]\!]^\dagger(S_0) R^+ [\![c_1]\!]^\dagger(S_1)$$

□

Example 3. In this example, we study the two commands

$$c_2 = \texttt{if}\ h = 0\ \texttt{then}\ l := 1\ \texttt{else}\ (h := 0) + (h := !h - 1)$$
$$c_3 = \texttt{if}\ h = 0\ \texttt{then}\ l := 1\ \texttt{else}\ (h := 0) + \texttt{skip}$$

which seem to share the same behavior on public variables, but that are inherently different because of their behavior on private variables. According to the semantics, we have:

$$\begin{array}{ll}
[\![c_2]\!](h \mapsto 0, l \mapsto _) & = \{(h \mapsto 0, l \mapsto 1)\}\downarrow \\
[\![c_2]\!](h \mapsto j+1, l \mapsto k) & = \{(h \mapsto j, l \mapsto k), (h \mapsto 0, l \mapsto k)\}\downarrow \\
[\![c_3]\!](h \mapsto 0, l \mapsto _) & = \{(h \mapsto 0, l \mapsto 1)\}\downarrow \\
[\![c_3]\!](h \mapsto j+1, l \mapsto k) & = \{(h \mapsto j+1, l \mapsto k), (h \mapsto 0, l \mapsto k)\}\downarrow
\end{array}$$

We can verify that $c_2 \preceq_R c_3$, with R defined as the smallest relation that satisfies our conditions (reflexivity, etc.) and such that

$$\{(h \mapsto k)\} R \{(h \mapsto k')\} \qquad \text{for all } k \leq k'$$

For example, let S_0 and S_1 be $\{(h \mapsto 5, l \mapsto 0)\}\downarrow$ and $\{(h \mapsto 7, l \mapsto 0)\}\downarrow$. Then we have $S_0 R^+ S_1$, and:

$$\begin{array}{l}
[\![c_2]\!]^\dagger(S_0) = \{(h \mapsto 4, l \mapsto 0), (h \mapsto 0, l \mapsto 0)\}\downarrow \\
[\![c_3]\!]^\dagger(S_1) = \{(h \mapsto 7, l \mapsto 0), (h \mapsto 0, l \mapsto 0)\}\downarrow
\end{array}$$

We can then check that:

$$[\![c_2]\!]^\dagger(S_0) R^+ [\![c_3]\!]^\dagger(S_1)$$

However there is no suitable relation R such that $c_3 \preceq_R c_2$. If there were such a relation R, it would be reflexive, so $\{(h \mapsto 1)\}\ R\ \{(h \mapsto 1)\}$. Suppose that $S_0 = \{(h \mapsto 1, l \mapsto 0)\}\downarrow$ and that $S_1 = \{(h \mapsto 1, l \mapsto 0)\}\downarrow$. We have $S_0 R^+ S_1$, and:

$$\begin{array}{l}
[\![c_3]\!]^\dagger(S_0) = \{(h \mapsto 1, l \mapsto 0), (h \mapsto 0, l \mapsto 0)\}\downarrow \\
[\![c_2]\!]^\dagger(S_1) = \{(h \mapsto 0, l \mapsto 0)\}\downarrow
\end{array}$$

We need

$$\{(h \mapsto 1, l \mapsto 0), (h \mapsto 0, l \mapsto 0)\}\downarrow\ R^+ \{(h \mapsto 0, l \mapsto 0)\}\downarrow$$

hence $\{(h \mapsto 1)\} R \{(h \mapsto 0)\}$. Now take $S_2 = \{(h \mapsto 1, l \mapsto 0)\}\downarrow$ and $S_3 = \{(h \mapsto 0, l \mapsto 0)\}\downarrow$. We have $S_2 R^+ S_3$, and:

$$\begin{array}{l}
[\![c_3]\!]^\dagger(S_2) = \{(h \mapsto 1, l \mapsto 0), (h \mapsto 0, l \mapsto 0)\}\downarrow \\
[\![c_2]\!]^\dagger(S_3) = \{(h \mapsto 0, l \mapsto 1)\}\downarrow
\end{array}$$

Since the values of l do not match, we cannot have $[\![c_3]\!]^\dagger(S_2) R^+ [\![c_2]\!]^\dagger(S_3)$, hence $c_3 \not\preceq_R c_2$.

As predicted by Theorem 1, we also have $c_3 \not\sqsubseteq_L c_2$. Indeed, for $C = \_\,;\_$ and $s_0 = (h \mapsto 1, l \mapsto 0)$, we have $[\![C[c_3]]\!](s_0) \not\leq_L [\![C[c_2]]\!](s_0)$. □

4 The Low-Level Language

In this section, we define our low-level language. In this language, we use concrete natural-number addresses for memory. We still use abstract location names, but those are interpreted as natural numbers (according to a memory layout), and can appear in arithmetic expressions.

4.1 Syntax and Informal Semantics

The syntax of the low-level language includes categories for natural-number expressions, boolean expressions, and commands:

$$
\begin{array}{rcl}
e & ::= & k \mid l_{\mathtt{nat}} \mid !e \mid e + e \mid e * e \\
b & ::= & e \leq e \mid \neg b \mid \mathrm{tt} \mid \mathrm{ff} \mid b \vee b \mid b \wedge b \\
c & ::= & e := e \mid \mathtt{if}\, b\, \mathtt{then}\, c\, \mathtt{else}\, c \mid \mathtt{skip} \mid c; c \mid c + c \mid \mathtt{while}\, b\, \mathtt{do}\, c
\end{array}
$$

where k ranges over numerals, and l over the finite set of store locations. Boolean expressions are as in the high-level language. Natural-number expressions and commands are also as in the high-level language, except for the inclusion of memory locations among the natural-number expressions, and for the dereferencing construct $!e$ and assignment construct $e := e'$ where e is an arbitrary natural-number expression (not necessarily a location).

Importantly, memory addresses are natural numbers, and a memory is a partial function from those addresses to contents. We assume that accessing an address at which the memory is undefined constitutes an error that stops execution immediately. In this respect, our language relies on the "fatal-error model" of Abadi and Plotkin [7]. With more work, it may be viable to treat also the alternative "recoverable-error model", which permits attacks to continue after such accesses, and therefore requires a bound on the number of such accesses.

4.2 Denotational Semantics

Low-Level Memories, Layouts, and Errors. We assume given a natural number $\mathrm{r} > |\mathrm{Loc}|$ that specifies the size of the memory. A memory m is a partial function from $\{1, \ldots, \mathrm{r}\}$ to the natural numbers; we write Mem for the set of memories. A memory layout w is an injection from Loc to $\{1, \ldots, \mathrm{r}\}$; we write $\mathrm{ran}(w)$ for its range. We consider only memory layouts that extend a given public memory layout w_p (an injection from PubLoc to $\{1, \ldots, \mathrm{r}\}$), fixed in the remaining of the paper. We let W be the set of those layouts.

The security of layout randomization depends on the randomization itself. We let d be a probability distribution on memory layouts (that extend w_p). When φ is a predicate on memory layouts, we write $P_d(\varphi(w))$ for the probability that $\varphi(w)$ holds with w sampled according to d.

Given a distribution d on layouts, we write δ_d for the minimum probability for a memory address to have no antecedent private location (much as in [7]):

$$\delta_d = \min_{i \in \{1,\ldots,\mathrm{r}\} \setminus \mathrm{ran}(w_p)} P_d(i \notin \mathrm{ran}(w))$$

We assume that $\delta_d > 0$. This assumption is reasonable, as $1 - \delta_d$ is the maximum probability for an adversary to guess a private location. For common distributions (e.g., the uniform distribution), δ_d approaches 1 as r grows, indicating that adversaries fail most of the time. We assume d fixed below, and may omit it, writing δ for δ_d.

The denotational semantics of the low-level language uses the "error + nontermination" monad $P_{\xi\perp} =_{\mathrm{def}} (P + \{\xi\})_\perp$, which first adds an "error" element ξ to P and then a least element. As the monad is strong, functions $f : P_1 \times \ldots \times P_n \to Q_{\xi\perp}$ extend to functions $\overline{f}$ on $(P_1)_{\xi\perp} \times \ldots \times (P_n)_{\xi\perp}$, where $\overline{f}(x_1, \ldots, x_n)$ is ξ or $\perp$ if some x_j, but no previous x_i, is; we often write f for $\overline{f}$.

For any memory layout w and store s, we let $w \cdot s$ be the memory defined on $\mathrm{ran}(w)$ by:

$$w \cdot s(i) = s(l) \text{ for } w(l) = i$$

(so that $w \cdot s(w(l)) = s(l)$). The notation $w \cdot s$ extends to $s \in S_{\xi\perp}$, as above, so that $w \cdot \xi = \xi$ and $w \cdot \perp = \perp$. A store projection is a function $\zeta : \mathrm{Mem}^W_{\xi\perp}$ of the form $w \mapsto w \cdot s$, for some $s \in S_{\xi\perp}$; we use the notation $- \cdot s$ to write such store projection functions.

What Should the Denotational Semantics be? A straightforward semantics might have type:

$$W \times \mathrm{Mem} \to \mathcal{H}(\mathrm{Mem}_{\xi\perp})$$

so that the meaning of a command would be a function from layouts and memories to sets of memories (modulo the use of the "error + nontermination" monad). Using a simple example we now argue that this is unsatisfactory, and arrive at a more satisfactory alternative.

Suppose that there is a unique private location l, no public locations, and that the memory has four addresses, $\{1, 2, 3, 4\}$. We write s_i for the

store $(l \mapsto i)$. The 4 possible layouts are $w_i = (l \mapsto i)$, for $i = 1, \ldots, 4$. Assume that d is uniform. Consider the following command:

$$c_4 \;=\; (1{:=}1) \;+\; (2{:=}1) \;+\; (3{:=}1) \;+\; (4{:=}1)$$

which nondeterministically guesses an address and attempts to write 1 into it. Intuitively, this command should fail to overwrite l most of the time. However, in a straightforward semantics of the above type we would have:

$$[\![c_4]\!](w_j, w_j{\cdot}s_0) = \{\xi, w_j{\cdot}s_1\}\downarrow$$

and we cannot state any quantitative property of the command, only that it sometimes fails and that it sometimes terminates.

One can rewrite the type of this semantics as:

$$\mathrm{Mem} \to \mathcal{H}(\mathrm{Mem}_{\xi\perp})^W$$

and view that as a type of functions that yield an $\mathcal{H}(\mathrm{Mem}_{\xi\perp})$-valued random variable with sample space W (the set of memory layouts) and distribution d. Thus, in this semantics, the nondeterministic choice is made after the probabilistic one —the wrong way around, as indicated in the Introduction.

It is therefore natural to reverse matters and look for a semantics of type:

$$\mathrm{Mem} \to \mathcal{H}(\mathrm{Mem}_{\xi\perp}^W)$$

now yielding a set of $\mathrm{Mem}_{\xi\perp}$-valued random variables—so, making the nondeterministic choice first. Desirable as this may be, there seems to be no good notion of composition of such functions.

Fortunately, this last problem can be overcome by changing the argument type to also be that of $\mathrm{Mem}_{\xi\perp}$-valued random variables:

$$\mathrm{Mem}_{\xi\perp}^W \to \mathcal{H}(\mathrm{Mem}_{\xi\perp}^W)$$

It turns out that with this semantics we have:

$$[\![c_4]\!](\zeta_i) = \{\zeta_\xi^1, \zeta_\xi^2, \zeta_\xi^3, \zeta_\xi^4\}\downarrow$$

where $\zeta_i(w) = w{\cdot}s_i$ and $\zeta_\xi^i(w) = w_i{\cdot}s_1$ if $w = w_i$ and $= \xi$ otherwise. We can then say that, for every nondeterministic choice, the probability of an error (or nontermination, as we are using the lower powerdomain) is 0.75.

In a further variant of the semantics, one might replace $\mathrm{Mem}_{\xi\perp}$-valued random variables by the corresponding probability distributions

on $\mathrm{Mem}_{\xi\perp}$, via the natural map $\mathrm{Ind}_d : \mathrm{Mem}_{\xi\perp}^W \longrightarrow \mathcal{V}(\mathrm{Mem}_{\xi\perp})$ induced by the distribution d on W (where $\mathcal{V}$ is the probabilistic powerdomain monad, see [13]). Such a semantics could have the form:

$$\mathrm{Mem} \to \mathcal{H}_{\mathcal{V}}(\mathrm{Mem}_{\xi\perp})$$

mapping memories to probability distributions on memories, where $\mathcal{H}_{\mathcal{V}}$ is a powerdomain for mixed nondeterministic and probabilistic choice as discussed above. However, such an approach would imply (incorrectly) that a new layout is chosen independently for each memory operation, rather than once and for all. In our small example with the single private location l and four addresses, it would not capture that $(1:=1);(2:=1)$ will always fail. It would treat the two assignments in $(1:=1);(2:=1)$ as two separate guesses that may both succeed. Similarly, it would treat the two assignments in $(1:=1);(1:=2)$ as two separate guesses where the second guess may fail to overwrite l even if the first one succeeds. With a layout chosen once and for all, on the other hand, the behavior of the second assignment is completely determined after the first assignment.

Denotational Semantics. The denotational semantics

$$[\![e]\!] : \mathrm{Mem} \times W \to \mathbb{N}_{\xi\perp} \qquad [\![b]\!] : \mathrm{Mem} \times W \to \mathbb{B}_{\xi\perp}$$

of expressions are defined in a standard way. In particular, $[\![l_{\mathtt{nat}}]\!]_m^w = w(l)$, and also $[\![!e]\!]_m^w = m([\![e]\!]_m^w)$, if $[\![e]\!]_m^w \in \mathrm{dom}(m)$, and $= \xi$, otherwise, using an obvious notation for functional application. Note that these semantics never have value $\perp$.

As discussed above, the denotational semantics of commands has type:

$$[\![c]\!] : \mathrm{Mem}_{\xi\perp}^W \to \mathcal{H}(\mathrm{Mem}_{\xi\perp}^W)$$

(and we remark that, as W is finite, all increasing chains in $\mathrm{Mem}_{\xi\perp}^W$ are eventually constant, and so for any nonempty subset X of $\mathrm{Mem}_{\xi\perp}^W$ we have $X^* = X\downarrow$). The denotational semantics is defined in Figure 3; it makes use of two auxiliary definitions. We first define:

$$\mathrm{Ass} : \mathrm{Mem}_{\xi\perp} \times \mathbb{N}_{\xi\perp} \times \mathbb{N}_{\xi\perp} \to \mathrm{Mem}_{\xi\perp}$$

by setting $\mathrm{Ass}(m,x,y) = m[x \mapsto y]$ if $x \in \mathrm{dom}(m)$ and $= \xi$, otherwise, for $m \in \mathrm{Mem}$, $x, y \in \mathbb{N}$, and then using the function extension associated to the "error + nontermination" monad. Second, we define

$$\mathrm{Cond}(p,\theta,\theta') : \mathrm{Mem}_{\xi\perp}^W \to \mathcal{H}(\mathrm{Mem}_{\xi\perp}^W)$$

for any $p : \mathrm{Mem} \times W \to \mathbb{B}_{\xi\perp}$ and $\theta, \theta' : \mathrm{Mem}^W_{\xi\perp} \to \mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$, by:

$$\mathrm{Cond}(p,\theta,\theta')(\zeta) = \{\zeta' \mid \zeta'|_{W_{\zeta,\mathrm{tt}}} \in \theta(\zeta)|_{W_{\zeta,\mathrm{tt}}}, \zeta'|_{W_{\zeta,\mathrm{ff}}} \in \theta'(\zeta)|_{W_{\zeta,\mathrm{ff}}}, \\ \zeta'(W_{\zeta,\xi}) \subseteq \{\xi\}, \text{ and } \zeta'(W_{\zeta,\perp}) \subseteq \{\perp\}\}\downarrow$$

where $W_{\zeta,t} =_{\mathrm{def}} \{w \mid p(\zeta(w), w) = t\}$, for $t \in \mathbb{B}_{\xi\perp}$, and we apply restriction elementwise to sets of functions.

$$\begin{array}{lll}
[\![c + c']\!](\zeta) = [\![c]\!](\zeta) \cup [\![c']\!](\zeta) & [\![c; c']\!] = [\![c']\!]^\dagger \circ [\![c]\!] & [\![\mathsf{skip}]\!] = \eta \\
[\![e := e']\!](\zeta) = \eta(\lambda w : W.\, \mathrm{Ass}(\zeta(w), [\![e]\!]^w_{\zeta(w)}, [\![e']\!]^w_{\zeta(w)})) & & \\
[\![\mathtt{if}\, b\, \mathtt{then}\, c\, \mathtt{else}\, c']\!] = \mathrm{Cond}([\![b]\!], [\![c]\!], [\![c']\!]) & & \\
[\![\mathtt{while}\, b\, \mathtt{do}\, c]\!] = \mu\theta : \mathrm{Mem}^W_{\xi\perp} \to \mathcal{H}(\mathrm{Mem}^W_{\xi\perp}).\mathrm{Cond}([\![b]\!], \theta^\dagger \circ [\![c]\!], \eta) & &
\end{array}$$

Fig. 3. Low-level denotational semantics

Example 4. In this example, we demonstrate our low-level denotational semantics. Consider the command:

$$c_5 = l'_{\mathtt{nat}} := l_{\mathtt{nat}}; (!l'_{\mathtt{nat}}) := 1; l'_{\mathtt{nat}} := 0$$

This command stores the address of location l at location l', then reads the contents of location l' (the address of l) and writes 1 at this address, and finally resets the memory at location l' to 0. Because of this manipulation of memory locations, this command is not the direct translation of a high-level command.

Letting:

$$s_{i,j} = (l \mapsto i, l' \mapsto j) \quad \zeta_{i,j} = -\cdot s_{i,j} \quad \zeta'_i = -\cdot(l \mapsto i, l' \mapsto w(l))$$

we have:

$$[\![l'_{\mathtt{nat}} := l_{\mathtt{nat}}]\!](\zeta_{i,j}) = \{\zeta'_i\}\downarrow$$

Note that $\zeta_{i,j}$ is a store projection, but ζ'_i is not. We also have:

$$[\![(!l'_{\mathtt{nat}}) := 1]\!](\zeta'_i) = \{\zeta'_1\}\downarrow \qquad [\![l'_{\mathtt{nat}} := 0]\!](\zeta'_1) = \{\zeta_{1,0}\}\downarrow$$

In sum, we have:

$$[\![c_5]\!](\zeta_{i,j}) = \{\zeta_{1,0}\}\downarrow$$

□

Looking at the type of the semantics

$$[\![c]\!] : \mathrm{Mem}^{W}_{\xi\perp} \to \mathcal{H}(\mathrm{Mem}^{W}_{\xi\perp})$$

one may be concerned that there is no apparent relation between the layouts used in the input to $[\![c]\!]$ and those in its output. However, we note that the semantics could be made parametric. For every $W' \subseteq W$, replace W by W' in the definition of $[\![c]\!]$ to obtain:

$$[\![c]\!]_{W'} : \mathrm{Mem}^{W'}_{\xi\perp} \to \mathcal{H}(\mathrm{Mem}^{W'}_{\xi\perp})$$

There is then a naturality property, that the following diagram commutes for all $W'' \subseteq W' \subseteq W$:

$$\begin{array}{ccc} \mathrm{Mem}^{W'}_{\xi\perp} & \xrightarrow{[\![c]\!]_{W'}} & \mathcal{H}(\mathrm{Mem}^{W'}_{\xi\perp}) \\ \Big\downarrow{\scriptstyle \mathrm{Mem}^{\iota}_{\xi\perp}} & & \Big\downarrow{\scriptstyle \mathcal{H}(\mathrm{Mem}^{\iota}_{\xi\perp})} \\ \mathrm{Mem}^{W''}_{\xi\perp} & \xrightarrow[[\![c]\!]_{W''}]{} & \mathcal{H}(\mathrm{Mem}^{W''}_{\xi\perp}) \end{array}$$

where $\iota : W'' \subseteq W'$ is the inclusion map. Taking $W' = W$ and W'' a singleton yields the expected relation between input and output: the value of a random variable in the output at a layout depends only on the value of the input random variable at that layout. The naturality property suggests re-working the low-level denotational semantics in the category of presheaves over sets of layouts, and this may prove illuminating (see [22] for relevant background).

4.3 Operational Semantics

As a counterpart to the denotational semantics, we give a big-step deterministic operational semantics using oracles to make choices.

The set of oracles Π is ranged over by π and is given by the following grammar:

$$\pi \ ::= \ \varepsilon \mid L\pi \mid R\pi \mid \pi;\pi \mid \mathit{if}(\pi,\pi)$$

A low-level state σ is:

- a pair $\langle c, m\rangle$ of a command c and a memory m,
- a memory m, or
- the error element ξ.

States of either of the last two forms are called terminal, and written τ. Transitions relate states and terminal states. They are given relative to a layout, and use an oracle to resolve nondeterminism. So we write:

$$w \models \sigma \stackrel{\pi}{\Rightarrow} \tau$$

Figure 4 gives the rules for this relation.

The rules for conditionals use different oracles for the true and false branches in order to avoid any correlation between the choices made in the two branches.[4] The rules for two commands in sequence also use different oracles, again avoiding correlations, now between the choices made in executing the first command and the choices made in executing the second. The oracles used in the rules for while loops ensure that the operational semantics of a loop and its unrolling are the same. We continue this discussion after Theorem 2 below.

Example 5. Consider the command c_4 introduced in Section 4.2, with added parentheses for disambiguation:

$$c_4 \;=\; (1{:=}1) \;+\; ((2{:=}1) \;+\; ((3{:=}1) \;+\; ((4{:=}1))))$$

We have:

$$\begin{array}{llll}
w_1 \models \langle c_4, w_1 \cdot s_k\rangle \stackrel{L}{\Rightarrow} w_1 \cdot s_1 & w_j \models \langle c_4, w_j \cdot s_k\rangle \stackrel{L}{\Rightarrow} \xi \; (j \neq 1) \\
w_2 \models \langle c_4, w_2 \cdot s_k\rangle \stackrel{RL}{\Longrightarrow} w_2 \cdot s_1 & w_j \models \langle c_4, w_j \cdot s_k\rangle \stackrel{RL}{\Longrightarrow} \xi \; (j \neq 2) \\
w_3 \models \langle c_4, w_3 \cdot s_k\rangle \stackrel{RRL}{\Longrightarrow} w_3 \cdot s_1 & w_j \models \langle c_4, w_j \cdot s_k\rangle \stackrel{RRL}{\Longrightarrow} \xi \; (j \neq 3) \\
w_4 \models \langle c_4, w_4 \cdot s_k\rangle \stackrel{RRR}{\Longrightarrow} w_4 \cdot s_1 & w_j \models \langle c_4, w_j \cdot s_k\rangle \stackrel{RRR}{\Longrightarrow} \xi \; (j \neq 4)
\end{array}$$

□

The transition relation is deterministic: if $w \models \sigma \stackrel{\pi}{\Rightarrow} \tau$ and $w \models \sigma \stackrel{\pi}{\Rightarrow} \tau'$ then $\tau = \tau'$. We can therefore define an evaluation function

$$\mathrm{Eval} : \mathrm{Com} \times W \times \mathrm{Mem} \times \Pi \to \mathrm{Mem}_{\xi\perp}$$

by:

$$\mathrm{Eval}(c, w, m, \pi) = \begin{cases} \tau \;\; (\text{if } w \models \langle c, m\rangle \stackrel{\pi}{\Rightarrow} \tau) \\ \perp \; (\text{otherwise}) \end{cases}$$

[4] The rules for, e.g., conditionals differ from those given in [17] which use the same oracle for both branches. The rules in [17] are erroneous in that the resulting operational semantics is not consistent with the denotational semantics in the sense of Theorem 2.

$$\frac{[\![e]\!]^w_m \in \mathrm{dom}(m) \text{ and } [\![e']\!]^w_m \neq \xi}{w \models \langle e := e', m\rangle \stackrel{\varepsilon}{\Rightarrow} m[[\![e]\!]^w_m \mapsto [\![e']\!]^w_m]} \qquad \frac{[\![e]\!]^w_m \notin \mathrm{dom}(m) \text{ or } [\![e']\!]^w_m = \xi}{w \models \langle e := e', m\rangle \stackrel{\varepsilon}{\Rightarrow} \xi}$$

$$\frac{[\![b]\!]^w_m = \mathrm{tt} \qquad \langle c, m\rangle \stackrel{\pi}{\Rightarrow} \tau}{w \models \langle \mathtt{if}\, b\, \mathtt{then}\, c\, \mathtt{else}\, c', m\rangle \xrightarrow{if(\pi,\pi')} \tau} \qquad \frac{[\![b]\!]^w_m = \mathrm{ff} \quad \langle c', m\rangle \stackrel{\pi'}{\Rightarrow} \tau}{w \models \langle \mathtt{if}\, b\, \mathtt{then}\, c\, \mathtt{else}\, c', m\rangle \xrightarrow{if(\pi,\pi')} \tau}$$

$$\frac{[\![b]\!]^w_m = \xi}{w \models \langle \mathtt{if}\, b\, \mathtt{then}\, c\, \mathtt{else}\, c', m\rangle \xrightarrow{if(\pi,\pi')} \xi} \qquad w \models \langle \mathsf{skip}, m\rangle \stackrel{\varepsilon}{\Rightarrow} m$$

$$\frac{w \models \langle c, m\rangle \stackrel{\pi}{\Rightarrow} m' \quad \langle c', m'\rangle \stackrel{\pi'}{\Rightarrow} \tau}{w \models \langle c; c', m\rangle \xrightarrow{\pi;\pi'} \tau} \qquad \frac{w \models \langle c, m\rangle \stackrel{\pi}{\Rightarrow} \xi}{w \models \langle c; c', m\rangle \xrightarrow{\pi;\pi'} \xi} \qquad \frac{w \models \langle c, m\rangle \stackrel{\pi}{\Rightarrow} \tau}{w \models \langle c + c', m\rangle \xrightarrow{L\pi} \tau}$$

$$\frac{w \models \langle c', m\rangle \stackrel{\pi}{\Rightarrow} \tau}{w \models \langle c + c', m\rangle \xrightarrow{R\pi} \tau} \qquad \frac{[\![b]\!]^w_m = \mathrm{ff}}{w \models \langle \mathtt{while}\, b\, \mathtt{do}\, c, m\rangle \xrightarrow{if(\pi;\pi',\varepsilon)} m}$$

$$\frac{[\![b]\!]^w_m = \mathrm{tt} \quad \langle c, m\rangle \stackrel{\pi}{\Rightarrow} m' \quad \langle \mathtt{while}\, b\, \mathtt{do}\, c, m'\rangle \stackrel{\pi'}{\Rightarrow} \tau}{w \models \langle \mathtt{while}\, b\, \mathtt{do}\, c, m\rangle \xrightarrow{if(\pi;\pi',\varepsilon)} \tau}$$

$$\frac{[\![b]\!]^w_m = \mathrm{tt} \quad \langle c, m\rangle \stackrel{\pi}{\Rightarrow} \xi}{w \models \langle \mathtt{while}\, b\, \mathtt{do}\, c, m\rangle \xrightarrow{if(\pi;\pi',\varepsilon)} \xi} \qquad \frac{[\![b]\!]^w_m = \xi}{w \models \langle \mathtt{while}\, b\, \mathtt{do}\, c, m\rangle \xrightarrow{if(\pi;\pi',\varepsilon)} \xi}$$

Fig. 4. Low-level operational semantics

In order to establish the consistency of the operational and denotational semantics we make use of an intermediate denotational semantics

$$[\![c]\!]_i : W \to (\mathrm{Mem} \times \Pi \to \mathrm{Mem}_{\xi\perp})$$

defined by setting for all commands, other than loops:

$$\begin{aligned}
[\![c + c']\!]_i(w)(m,\pi) &= \begin{cases} [\![c]\!]_i(w)(m,\pi') & (\pi = L\pi') \\ [\![c']\!]_i(w)(m,\pi') & (\pi = R\pi') \end{cases} \\
[\![c; c']\!]_i(w)(m,\pi;\pi') &= \overline{[\![c']\!]_i(w)}([\![c]\!]_i(w)(m,\pi),\pi') \\
[\![\mathsf{skip}]\!]_i(w)(m,\varepsilon) &= m \\
[\![e := e']\!]_i(w)(m,\varepsilon) &= \mathrm{Ass}(w, [\![e]\!]^w_m, [\![e']\!]^w_m) \\
[\![\mathtt{if}\, b\, \mathtt{then}\, c\, \mathtt{else}\, c']\!]_i(w)(m, if(\pi;\pi')) &= \mathrm{C}([\![b]\!]^w_m, [\![c]\!]_i(w)(m,\pi), [\![c']\!]_i(w)(m,\pi'))
\end{aligned}$$

and taking $[\![c]\!]_i(w)(m,\pi)$ to be $\perp$ for all other combinations of loop-free commands and oracles, and where we use the error plus nontermination extension of the evident conditional function $\mathrm{C} : \mathbb{B} \times P \times P \to P$ for the semantics of conditionals.

For while loops, $[\![\texttt{while}\, b\, \texttt{do}\, c]\!]_i$ is defined to be

$$\mu\theta.\lambda w.\lambda m,\pi.\begin{cases}\mathrm{C}([\![b]\!]^w_m, \overline{\theta(w)}([\![c]\!]_i(w)(m,\pi'),\pi''),m) & (\pi = if(\pi';\pi'',\varepsilon))\\ \bot & (\text{otherwise})\end{cases}$$

The following lemma asserts the consistency of the operational semantics and this intermediate denotational semantics. Its proof, which uses rule-induction, structural induction, and consideration of iterates like that of Proposition 1, is omitted.

Lemma 4. *For any low-level command c, layout w, memory m, and oracle π, we have:*

$$[\![c]\!]_i(w)(m,\pi) = \mathrm{Eval}(c,w,m,\pi)$$

Lemma 5. *For any low-level command c and $\zeta \in \mathrm{Mem}^W_{\xi\bot}$, we have:*

$$[\![c]\!](\zeta) = \{\lambda w{:}W.\, \overline{[\![c]\!]_i(w)}(\zeta(w),\pi) \mid \pi \in \Pi\}\downarrow$$

Proof. We first extend the language by adding a command Ω, and let it denote the relevant least element in both denotational semantics. With that we establish the result for commands not containing any while loops, proceeding by structural induction:

1. skip, Ω*:* These two cases are immediate from the definitions of $[\![c]\!]$ and $[\![c]\!]_i$.
2. $e := e'$: We calculate:

$$\begin{aligned}\{\lambda w{:}W.\, \overline{[\![e:=e']\!]_i(w)}(\zeta(w),\pi) \mid \pi\in\Pi\}\downarrow &\\ = \{\lambda w{:}W.\, \overline{[\![e:=e']\!]_i(w)}(\zeta(w),\varepsilon)\}\downarrow &\\ = \{\lambda w{:}W.\, \mathrm{Ass}(w,[\![e]\!]^w_{\zeta(w)},[\![e']\!]^w_{\zeta(w)})\}\downarrow &\\ = [\![e:=e']\!](\zeta)&\end{aligned}$$

3. $c; c'$: We calculate:

$$\begin{aligned}[\![c;c']\!](\zeta) &= \bigcup\{[\![c']\!](\zeta') \mid \zeta' \in [\![c]\!](\zeta)\}\downarrow\\ &= \bigcup\{[\![c']\!](\lambda w{:}W.\, \overline{[\![c]\!]_i(w)}(\zeta(w),\pi)) \mid \pi\in\Pi\}\downarrow\\ &= \bigcup\{\{\lambda w{:}W.\, \overline{[\![c']\!]_i(w)}(\overline{[\![c]\!]_i(w)}(\zeta(w),\pi),\pi') \mid \pi'\in\Pi\downarrow\}\\ &\qquad \mid \pi\in\Pi\}\downarrow\\ &= \{\lambda w{:}W.\, \overline{[\![c']\!]_i(w)}(\overline{[\![c]\!]_i(w)}(\zeta(w),\pi),\pi') \mid \pi,\pi'\in\Pi\}\downarrow\\ &= \{\lambda w{:}W.\, \overline{[\![c;c']\!]_i(w)}(\zeta(w),(\pi;\pi')) \mid \pi,\pi'\in\Pi\}\downarrow\\ &= \{\lambda w{:}W.\, \overline{[\![c;c']\!]_i(w)}(\zeta(w),\pi) \mid \pi\in\Pi\}\downarrow\end{aligned}$$

4. $c_L + c_R$: We calculate:

$$\begin{aligned}\{\lambda w{:}W.\,\overline{[\![c_L + c_R]\!]_i(w)}(\zeta(w),\pi) \mid \pi \in \Pi\}\downarrow \quad &= \\ \{\lambda w{:}W.\,\overline{[\![c_L + c_R]\!]_i(w)}(\zeta(w),L\pi) \mid \pi \in \Pi\}\downarrow & \\ \cup\, \{\lambda w{:}W.\,\overline{[\![c_L + c_R]\!]_i(w)}(\zeta(w),R\pi) \mid \pi \in \Pi\}\downarrow & \\ = & \\ \{\lambda w{:}W.\,\overline{[\![c_L]\!]_i(w)}(\zeta(w),\pi) \mid \pi \in \Pi\}\downarrow & \\ \cup\, \{\lambda w{:}W.\,\overline{[\![c_R]\!]_i(w)}(\zeta(w),\pi) \mid \pi \in \Pi\}\downarrow & \\ = [\![c_L]\!](\zeta) \cup [\![c_R]\!](\zeta) & \\ = [\![c_L + c_R]\!](\zeta) &\end{aligned}$$

5. $\mathtt{if}\, b\, \mathtt{then}\, c_{\mathrm{tt}}\, \mathtt{else}\, c_{\mathrm{ff}}$: We have to show that

$$\begin{aligned}[\![\mathtt{if}\, b\, \mathtt{then}\, c_{\mathrm{tt}}\, \mathtt{else}\, c_{\mathrm{ff}}]\!](\zeta) \quad &= \\ \{\lambda w{:}W.\,\overline{[\![\mathtt{if}\, b\, \mathtt{then}\, c_{\mathrm{tt}}\, \mathtt{else}\, c_{\mathrm{ff}}]\!]_i(w)}(\zeta(w),\pi) \mid \pi \in \Pi\}\downarrow &\end{aligned}$$

Set $W_{\zeta,t} =_{\mathrm{def}} \{w \mid [\![b]\!]^w_{\zeta(w)} = t\}$, for $t \in \mathbb{B}_{\xi\perp}$. Then note first that ζ' is in the left-hand side if, and only if, there are $\zeta'' \geq \zeta'$, $\zeta_{\mathrm{tt}} \in [\![c_{\mathrm{tt}}]\!](\zeta)$, and $\zeta_{\mathrm{ff}} \in [\![c_{\mathrm{ff}}]\!](\zeta)$ s.t.: $\zeta''|_{W_{\zeta,\mathrm{tt}}} = \zeta_{\mathrm{tt}}|_{W_{\zeta,\mathrm{tt}}}, \zeta''|_{W_{\zeta,\mathrm{ff}}} = \zeta_{\mathrm{ff}}|_{W_{\zeta,\mathrm{ff}}}, \zeta''(W_{\zeta,\xi}) \subseteq \{\xi\}$, and $\zeta''(W_{\zeta,\perp}) \subseteq \{\perp\}$.
Using the induction hypothesis, we see that $\zeta_{\mathrm{tt}} \in [\![c_{\mathrm{tt}}]\!](\zeta)$ if, and only if, for some π_{tt}, $\zeta_{\mathrm{tt}} \leq \lambda w{:}W.\,\overline{[\![c_{\mathrm{tt}}]\!]_i(w)}(\zeta(w),\pi_{\mathrm{tt}})$, and similarly for ζ_{ff}. We then see that the condition for ζ' to be in the left-hand side is equivalent to the existence of $\zeta'' \geq \zeta'$, π_{tt} and π_{ff} such that

$$\zeta'' \leq \lambda w{:}W.\,\mathrm{C}([\![b]\!]^w_{\zeta(w)}, [\![c_{\mathrm{tt}}]\!]_i(w)(\zeta(w),\pi_{\mathrm{tt}}), [\![c_{\mathrm{ff}}]\!]_i(w)(\zeta(w),\pi_{\mathrm{ff}}))$$

which is equivalent to the condition that ζ' is in the right-hand side.

We can now establish the desired result for general commands c (including Ω). Define iterates $c^{(n)}$ by setting $c^{(0)} = \Omega$ and defining $c^{(n+1)}$ homomorphically, except for while loops, where we put:

$$(\mathtt{while}\, b\, \mathtt{do}\, c_w)^{(n+1)} = \mathtt{if}\, b\, \mathtt{then}\, c_w^{(n+1)}; (\mathtt{while}\, b\, \mathtt{do}\, c_w)^{(n)}\, \mathtt{else}\, \mathsf{skip}$$

Note that the iterates are all in the sub-language not including loops.

We have that $[\![c^{(n)}]\!]$ is an increasing sequence with lub $[\![c]\!]$, and the same holds for $[\![-]\!]_i$. As the desired result holds for commands not con-

taining while loops, but possibly containing Ω, we can then calculate:

$$\begin{aligned}
[\![c]\!](\zeta) &= \bigvee_{n\geq 0} [\![c^{(n)}]\!](\zeta) \\
&= \bigvee_{n\geq 0} \{\lambda w{:}W.\, \overline{[\![c^{(n)}]\!]_i(w)}(\zeta(w),\pi) \mid \pi \in \Pi\}\downarrow \\
&= \{\lambda w{:}W.\, \overline{[\![c^{(n)}]\!]_i(w)}(\zeta(w),\pi) \mid \pi \in \Pi, n \geq 0\}\downarrow \\
&= \{\lambda w{:}W.\, \overline{\bigvee_{n\geq 0} [\![c^{(n)}]\!]_i(w)}(\zeta(w),\pi) \mid \pi \in \Pi\}\downarrow \\
&= \{\lambda w{:}W.\, \overline{\bigvee_{n\geq 0} [\![c^{(n)}]\!]_i(w)}(\zeta(w),\pi) \mid \pi \in \Pi\}\downarrow \\
&= \{\lambda w{:}W.\, \overline{[\![c]\!]_i(w)}(\zeta(w),\pi) \mid \pi \in \Pi\}\downarrow
\end{aligned}$$

which concludes the proof.

Lemmas 4 and 5 immediately yield the consistency of the operational and denotational semantics:

Theorem 2 (Low-level operational/denotational consistency). *For any low-level command c and $\zeta \in \mathrm{Mem}_{\xi\perp}^{W}$, we have:*

$$[\![c]\!](\zeta) = \{\lambda w{:}W.\, \overline{\mathrm{Eval}}(c, w, \zeta(w), \pi) \mid \pi \in \Pi\}\downarrow$$

The evaluation function yields operational correlates of the other possible denotational semantics discussed in Section 4.2, similarly, using image or induced distribution functionals. For example, for the first of those semantics, by currying Eval and composing, one obtains:

$$\mathrm{Com} \times W \times \mathrm{Mem} \xrightarrow{\mathrm{curry(Eval)}} \mathrm{Mem}_{\xi\perp}^{\Pi} \xrightarrow{\mathrm{Im}_{\mathrm{Mem}_{\xi\perp}}} \mathcal{P}(\mathrm{Mem}_{\xi\perp})$$

Using such operational correlates, one can verify operational versions of the assertions made in Section 4.2 about the inadequacies of those semantics.

The operational semantics has the peculiarity that the oracles used are independent of the layout but not of the command structure. Allowing the oracle to depend on the layout would amount to making nondeterministic choices after probabilistic ones. The use of the syntactic dependence in the case of conditionals can be seen by considering the example:

$$\mathtt{if}\ b\ \mathtt{then}\ (l_{\mathtt{nat}} := 0) + (l_{\mathtt{nat}} := 1)\ \mathtt{else}\ (l_{\mathtt{nat}} := 0) + (l_{\mathtt{nat}} := 1)$$

If the same oracle was used for both branches in the operational semantics, then $l_{\mathtt{nat}}$ would either always (i.e., for all layouts) be set to 0 or else would always be set to 1; however, for a suitable choice of condition b, the denotational semantics allows the possibility of setting $l_{\mathtt{nat}}$ to 0 in some layouts and to 1 in others.

In the case of sequence commands, consider the example

$$(\texttt{if}\ b\ \texttt{then skip else skip}); ((l_{\texttt{nat}} := 0) + (l_{\texttt{nat}} := 1))$$

If the oracle chosen for the second command depended on which branch of the conditional was taken, then it could be possible that $l_{\texttt{nat}}$ was sometimes set to 0 and sometimes to 1, whereas to be in accord with the denotational semantics it should either always be set to 0 or always set to 1.

In this connection it is worth noting that the equation

$$(\texttt{if}\ b\ \texttt{then}\ c_{\texttt{tt}}\ \texttt{else}\ c_{\texttt{ff}}); c = \texttt{if}\ b\ \texttt{then}\ c_{\texttt{tt}}; c\ \texttt{else}\ c_{\texttt{ff}}; c$$

which one might naturally expect to hold in the denotational semantics in fact does not. A counterexample can be obtained by taking $c_{\texttt{tt}}$ and $c_{\texttt{ff}}$ to be skip and c to be $(l_{\texttt{nat}} := 0) + (l_{\texttt{nat}} := 1)$. The left-hand side is then the command just considered to illustrate the use of oracles for sequence commands, and the right-hand side may sometimes set $l_{\texttt{nat}}$ to 0 and sometimes to 1.

Such subtleties, and more generally the difficulty of both operational and denotational semantics, suggest that these semantics may be attractive subjects for further work. Fortunately, neither the operational semantics nor its relation with the denotational semantics are needed for our main results (which are also those of [17]).

4.4 Implementation Relations and Equivalences

Much as in the high-level language, we define a contextual implementation relation and a simulation relation for the low-level language. The low-level definitions refer to layouts, and in some cases include conditions on induced probabilities.

Contextual Pre-order. Again, the contextual pre-order $c \sqsubseteq_L c'$ may be interpreted as saying that c "refines" (or "implements") c', in the sense that the publicly observable outcomes that c can produce are a subset of those that c' permits, in every public context. In comparison with the definition for the high-level language, however, c and c' are not applied to an arbitrary initial store but rather to a function from layouts to memories (extended with "error + nontermination"), and they produce sets of such functions. We restrict attention to argument functions induced by stores, in the sense that they are store projections of the form $- \cdot s$. Thus, let

$f = [\![C[c]]\!]$ and $f' = [\![C[c']]\!]$ for an arbitrary public context C, and let s be a store; then (roughly) for every ζ in $f(-\cdot s)$ there exists ζ' in $f'(-\cdot s)$ such that, for any w, $\zeta(w)$ and $\zeta'(w)$ coincide on public locations.

The treatment of error and nontermination introduces a further complication. Specifically, we allow that ζ produces an error or diverges with sufficient probability ($\geq \delta$), and that ζ' produces an error with sufficient probability ($\geq \delta$), as an alternative to coinciding on public locations.

Therefore, we define $\sqsubseteq_L$ and some auxiliary notation and relations:

- Set $\mathrm{PubMem} =_{\mathrm{def}} \mathbb{N}^{\mathrm{ran}(w_p)}$. Then, for any memory m, let m_L in PubMem be the restriction of m to $\mathrm{ran}(w_p)$, extending the notation to $\mathrm{Mem}_{\xi\perp}$ as usual.
- For any $\zeta \in \mathrm{Mem}^W_{\xi\perp}$, we define $\zeta_L \in \mathrm{PubMem}^W_{\xi\perp}$ by $\zeta_L(w) = \zeta(w)_L$.
- For $X, Y \in \mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$, we write that $X \leq_L Y$ when, for every $\zeta \in X$, there exists $\zeta' \in Y$ such that:
  - $\zeta_L \leq \zeta'_L$, or
  - $P(\zeta(w) \in \{\xi, \perp\}) \geq \delta$ and $P(\zeta'(w) = \xi) \geq \delta$.
- For $f, f' \in \mathrm{Mem}^W_{\xi\perp} \to \mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$, we write $f \leq_L f'$ when, for all $s \in S$, we have:
$$f(-\cdot s) \leq_L f'(-\cdot s)$$
- Finally, we write $c \sqsubseteq_L c'$ when, for every public command context C, $[\![C[c]]\!] \leq_L [\![C[c']]\!]$.

Simulation. As in the high-level language, we introduce a simulation relation $\preceq$. This relation works only on commands whose outcomes on inputs that are store projections are themselves store projections; nevertheless, simulation remains a useful tool for proofs.

We first define some auxiliary notations:

- We define $\max(X)$, for any $X \in \mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$, as the set of maximal elements of X. (As W is finite, every element of X is less than a maximal element of X, and $X = \max(X)\downarrow$.)
- We write $S(\zeta)$ for the element of $S_{\xi\perp}$ uniquely determined by a store projection ζ.
- For any cpo P and $\zeta \in P^W_{\xi\perp}$, we define $\zeta_{/\xi}$ by:
$$\zeta_{/\xi} = \begin{cases} w \mapsto \xi & (\text{if } P(\zeta(w) = \xi) \geq \delta) \\ \zeta & (\text{otherwise}) \end{cases}$$

– For every $X \in \mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$, we say that X is a store projection set when $\{\zeta_{/\xi} \mid \zeta \in \max(X)\}$ is a set of store projections. Then, we let

$$\chi(X) = S(\{\zeta_{/\xi} \mid \zeta \in \max(X)\}) \cup \{\perp\}$$

Note that $s \in \chi(X)$ if, and only if, $- \cdot s \in X$, and that $\xi \in \chi(X)$ if, and only if, there exists $\zeta \in X$ such that $P(\zeta(w) = \xi) \geq \delta$.

The $\leq_L$ relation restricted to store projection sets has a pleasant characterization. This characterization requires some definitions. First, $-_L$ extends from S to $S_{\xi\perp}$, so that $\perp_L = \perp$ and $\xi_L = \xi$; with that, for any X in $\mathcal{H}(S_{\xi\perp})$, we define X_L in $\mathcal{H}(S_{L\xi\perp})$ to be $\{s_L \mid s \in X\}$.

Fact 3 *Let X and Y be store projection sets. Then:*

$$X \leq_L Y \quad \Leftrightarrow \quad \chi(X)_L \leq \chi(Y)_L$$

Proof. Let X and Y be store projection sets. Assume first that $X \leq_L Y$, and take a non-bottom element of $\chi(X)_L$. There are two cases. In the first the element is s_L for some $s \in \mathrm{Store}$ such that $\zeta =_{\mathrm{def}} - \cdot s \in X$. As $X \leq_L Y$ we have $\zeta_L \leq \zeta'_L$ for some $\zeta' \in Y$. But then $\zeta' = - \cdot s'$ for some $s' \in \mathrm{Store}$ with $s'_L = s_L$ and so $s_L \in \chi(Y)_L$. In the second case the element is ξ and there is a $\zeta \in X$ such that $P(\zeta(w) = \xi) \geq \delta$. As $X \leq_L Y$ it follows that there is a $\zeta' \in Y$ such that $P(\zeta'(w) = \xi) \geq \delta$, and so $\xi \in \chi(Y)_L$.

For the converse, assume that $\chi(X)_L \leq \chi(Y)_L$. The case $X = \{\perp\}$ is trivial. Otherwise take $\zeta \in X$. Choose $\zeta' \in \max(X)$ such that $\zeta \leq \zeta'$. As X is a store projection set $\neq \{\perp\}$, there are two cases. In the first case ζ' has the form $- \cdot s$. As $\chi(X)_L \leq \chi(Y)_L$ there is a $\zeta'' \in Y$ of the form $- \cdot s'$ where $s'_L = s_L$. We therefore have $\zeta_L \leq \zeta'_L$. In the second case $P(\zeta'(w) = \xi) \geq \delta$ and so $\xi \in \chi(X)$ (and $P(\zeta(w) \in \{\xi, \perp\}) \geq \delta$). As $\chi(X)_L \leq \chi(Y)_L$ it follows that $\xi \in \chi(Y)$, and so there is a $\zeta'' \in Y$ such that $P(\zeta''(w) = \xi) \geq \delta$, which concludes the proof.

Much as in the high-level language, we extend relations R over $\mathcal{H}(S_{H\xi\perp})$ to relations $R^\times$ over $\mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$. First we extend $-_s$ to $\mathcal{H}(S_{\xi\perp})$ as follows: for $X \in \mathcal{H}(S_{\xi\perp})$ and $s \in S_L$, we let $X_s \in \mathcal{H}(S_{H\xi\perp})$ be $(X \setminus \{\xi\})_s \cup \{\xi \mid \xi \in X\}$. Then, given a relation R over $\mathcal{H}(S_{H\xi\perp})$, we first extend it to a relation R^+ over $\mathcal{H}(S_{\xi\perp})$ by setting

$$\begin{aligned} X R^+ Y \equiv_{\mathrm{def}} & (\xi \in X \Rightarrow \xi \in Y) \wedge \\ & \forall s \in S_L.\, ((X_s \setminus \{\xi\}) \neq \{\perp\} \Rightarrow (Y_s \setminus \{\xi\}) \neq \{\perp\}) \wedge X_s R Y_s \end{aligned}$$

for $X, Y \in \mathcal{H}(S_{\xi\perp})$ and then define $R^\times$ by setting:

$$X R^\times Y \equiv_{\text{def}} X \text{ and } Y \text{ are store projection sets} \wedge \chi(X) R^+ \chi(Y)$$

for $X, Y \in \mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$. (Note that if $R \subseteq \mathcal{H}(S_{H\perp})$, then the high- and low-level definitions of R^+ coincide.)

If R is closed under increasing ω-sups (respectively, is right-closed under $\leq$, is closed under binary unions) the same holds for R^+, and then for $R^\times$ (with $\leq$ restricted to store projection sets). If R is reflexive, then R^+ is and $R^\times$ is reflexive on store projection sets. We also have, much as before, that, for $X, Y \in \mathcal{H}(S_{\xi\perp})$, if $X R^+ Y$ then $X_L \leq Y_L$. It then follows from Fact 3 that, for $X, Y \in \mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$, if $X R^\times Y$ then $X \leq_L Y$.

For any $f, f' : \mathrm{Mem}^W_{\xi\perp} \to \mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$ and relation R over $\mathcal{H}(S_{H\perp})$ we write that $f \preceq_R f'$ when:

$$\forall X, Y \in \mathcal{H}(\mathrm{Mem}^W_{\xi\perp}).\, X R^\times Y \Rightarrow f^\dagger(X) R^\times f'^\dagger(Y)$$

Finally, we write that $f \preceq f'$ if $f \preceq_R f'$ for some reflexive R closed under increasing ω-sups, right-closed under $\leq$, and closed under binary unions.

Contextual Pre-order vs. Simulation. The contextual pre-order coincides with the simulation relation, but only for commands whose semantics sends store projections to store projection sets. Formally, we say that a given function $f : \mathrm{Mem}^W_{\xi\perp} \to \mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$ preserves store projections if, for every $s \in S$, $f(- \cdot s)$ is a store projection set. The coincidence remains quite useful despite this restriction, which in particular is not an impediment to our overall goal of relating the low-level language to the high-level language.

As in Section 3 we divide the proof of the coincidence into two halves. First, however, we need some preliminary lemmas.

Lemma 6. *For any cpo P, $\zeta \in \mathrm{Mem}^W_{\xi\perp}$, layout w, expression e, boolean expression b, command c, and $x \in \{\perp, \xi\}$ we have:*

$$\begin{aligned}
&\zeta(w) = x \Rightarrow [\![e]\!]^w_{\zeta(w)} = x \\
&\zeta(w) = x \Rightarrow [\![b]\!]^w_{\zeta(w)} = x \\
&\zeta(w) = x \Rightarrow \zeta'(w) = x \quad (\zeta' \in \max([\![c]\!](\zeta)))
\end{aligned}$$

Proof. For expressions and boolean expressions, the proof is by definition. The proof for commands is then straightforward by structural induction.

Lemma 7. *Let $f : \mathrm{Mem}^W_{\xi\perp} \to \mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$ be a function that preserves store projections. Let $X \in \mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$ be a store projection set. Then $f^\dagger(X)$ is a store projection set.*

Proof. We know that $f^\dagger(X) = \{f(\zeta) | \zeta \in \max(X)\}\downarrow$. We need to prove that, for all $\zeta \in \max(X)$ and $\zeta' \in \max(f(\zeta))$, we have that $\zeta'_{/\xi}$ is a store projection. Since X is a store projection set, we know that $\zeta_{/\xi}$ is a store projection, so we have three possibilities:

- If ζ is a store projection, we conclude since f preserves store projection sets.
- If $\zeta = \perp$, or $\zeta_{/\xi} = \xi$, we conclude using Lemma 6.

Lemma 8. *1. Suppose that e is a public natural-number expression, and that $\zeta \in \mathrm{Mem}^W_{\xi\perp}$ is such that $\zeta_{/\xi}$ is a store projection. Then either:*
   - $(w \mapsto [\![e]\!]^w_{\zeta(w)})_{/\xi} = w \mapsto \xi$,
   - $(w \mapsto [\![e]\!]^w_{\zeta(w)})_{/\xi} = \perp$ *and* $\zeta_{/\xi} = \perp$, *or*
   - *there exists* $n \in \mathbb{N}$ *such that* $(w \mapsto [\![e]\!]^w_{\zeta(w)})_{/\xi} = w \mapsto n$.

   Further, $(w \mapsto [\![e]\!]^w_{\zeta(w)})_{/\xi}$ *only depends on* $S(\zeta_{/\xi})_L$.

2. Suppose that b is a public boolean expression, and that $\zeta \in \mathrm{Mem}^W_{\xi\perp}$ is such that $\zeta_{/\xi}$ is a store projection. Then either:
   - $(w \mapsto [\![b]\!]^w_{\zeta(w)})_{/\xi} = w \mapsto \xi$,
   - $(w \mapsto [\![b]\!]^w_{\zeta(w)})_{/\xi} = \perp$ *and* $\zeta_{/\xi} = \perp$, *or*
   - *there exists* $t \in \mathbb{B}$ *such that* $(w \mapsto [\![b]\!]^w_{\zeta(w)})_{/\xi} = t$.

   Further, $(w \mapsto [\![b]\!]^w_{\zeta(w)})_{/\xi}$ *only depends on* $S(\zeta_{/\xi})_L$.

Proof. For the first part, letting e be a public expression, and letting $\zeta \in W \to \mathrm{Mem}_{\xi\perp}$ be such that $\zeta_{/\xi}$ is a store projection, if $S(\zeta_{/\xi}) = \xi$ or $S(\zeta_{/\xi}) = \perp$, we conclude, using Lemma 6. Otherwise $S(\zeta_{/\xi}) \in S$ and the proof is by structural induction on e. Note that as $S(\zeta_{/\xi}) \in S$, $[\![e']\!]^w_{\zeta(w)} \neq \perp$ for any e' and w, and so the second case cannot arise when applying the induction hypothesis.

1. k: The conclusion is immediate.
2. $l_{\mathtt{nat}}$: Since $l_{\mathtt{nat}}$ is public, and w_p is fixed, $[\![e]\!]^w_{\zeta(w)} = w_p(l)$ holds for every layout w, and we conclude.
3. $!e$: By the induction hypothesis on e, $(w \mapsto [\![e]\!]^w_{\zeta(w)})_{/\xi}$ only depends on $S(\zeta_{/\xi})_L$ and either
   - $P([\![e]\!]^w_{\zeta(w)} = \xi) \geq \delta$, in which case $P([\![!e]\!]^w_{\zeta(w)} = \xi) \geq \delta$ and we conclude

   or

- there exists an $n \in \mathbb{N}$ such that, for every layout w, $[\![e]\!]^w_{\zeta(w)} = n$. If $n = w_p(l)$ for some public l, then $[\![!e]\!]^w_{\zeta(w)} = \zeta(w)(l) = S(\zeta_{/\xi})(l)$, and we conclude. Otherwise, if $n \notin \mathrm{ran}(w_p)$, then we have that $P([\![!e]\!]^w_{\zeta(w)} = \xi) = P(n \notin \mathrm{ran}(w)) \geq \delta$, and we conclude.

4. $e + e$ or $e * e$: We conclude by the induction hypothesis.

The proof of the second part is similar; the corresponding induction makes use of the first part in the case when b has the form $e \leq e'$.

We can now prove the first half of the coincidence.

Lemma 9. *Let c and c' be two commands of the low-level language such that $[\![c]\!] \preceq [\![c']\!]$. Then $c \sqsubseteq_L c'$ holds.*

Proof. Let c_0 and c_1 be two commands such that $[\![c_0]\!] \preceq_R [\![c_1]\!]$, with R a reflexive relation closed under increasing ω-sups, right-closed under $\leq$, and closed under binary unions, and let C be a public command context.

We prove by induction on the size of C that $[\![C[c_0]]\!] \preceq_R [\![C[c_1]]\!]$, considering the possible forms of C:

1. $e := e'$: Suppose that $XR^{\times}Y$. We first do a case study on the semantics of e, e', and $e := e'$. As e and e' are public, for every $\zeta \in \mathrm{Mem}^W_{\xi\perp}$ such that $\zeta_{/\xi}$ is a store projection, we have, by Lemma 8
   - $(w \mapsto [\![e]\!]^w_{\zeta(w)})_{/\xi} = \xi$,
   - $(w \mapsto [\![e]\!]^w_{\zeta(w)})_{/\xi} = \perp$ and $\zeta = \perp$, or
   - there exists $n \in \mathbb{N}$ such that $(w \mapsto [\![e]\!]^w_{\zeta(w)})_{/\xi} = w \mapsto n$.

   and similarly for e'.
   In the bottom case, we have $\zeta = \perp$. In any of the error cases, we have $P([\![e := e']\!]^w_{\zeta(w)} = \xi) \geq \delta$, hence $\chi([\![e := e']\!](\zeta)) = \{w \mapsto \xi\}\downarrow$.
   Otherwise, let n and n' be such that $(w \mapsto [\![e]\!]^w_{\zeta(w)})_{/\xi} = w \mapsto n$ and $(w \mapsto [\![e']\!]^w_{\zeta(w)})_{/\xi} = w \mapsto n'$. By Lemma 6, and definition of $()_{/\xi}$ and store projections, there exists $s \in S$ such that $\zeta = - \cdot s$
   - If $n = w_p(l)$ for some public l, then we have:

     $$[\![e := e']\!](\zeta) = \{- \cdot s[l \mapsto n']\}\downarrow$$

     We then say that ζ is normal, and write $s^+(\zeta)$ for $(s[l \mapsto n'])_L$.
   - Otherwise, $n \notin \mathrm{ran}(w_p)$, and we have:

     $$P([\![e := e']\!](\zeta)(w) = \xi) = P(n \notin \mathrm{ran}(w)) \geq \delta$$

By Lemma 8, this analysis only depends on $S(\zeta_{/\xi})_L$.
We now prove that $[\![e := e']\!](X) R^{\times} [\![e := e']\!](Y)$. From the case analysis above, we deduce that $[\![e := e']\!](X)$ and $[\![e := e']\!](Y)$ are store projection sets. Also, $\xi \in \chi([\![e := e']\!](X))$ if and only if either there exists $\zeta \in X$ such that $P(\zeta(w) = \xi) \geq \delta$ or else there exists $- \cdot s \in X$ such that $\xi \in \chi([\![e{:=}e']\!](- \cdot s'))$ if $s'_L = s_L$ and similarly for Y. Since $X R^{\times} Y$, this proves that $\xi \in \chi([\![e{:=}e']\!](X)) \Rightarrow \xi \in \chi([\![e{:=}e']\!](Y))$.
Further, as can be seen from our case analysis above, for every $s' \in S_L$,

$$\begin{aligned}\chi([\![e{:=}e']\!](X))_{s'} = & \textstyle\bigcup_{\zeta \in \max(X)} \{\chi(X)_{S(\zeta_{/\xi})_L} \mid \zeta \text{ is normal}, s' = s^+(\zeta)\} \\ & \cup \{\xi \mid \exists \zeta \in \max(X).\, \zeta \text{ is not normal}, \zeta \neq \bot\} \\ & \cup \{\bot\}\end{aligned}$$

and similarly for Y. By hypothesis, $X R^{\times} Y$, and so we have both $X \leq_L Y$ (by Fact 3) and $\chi(X) R^{+} \chi(Y)$. From the latter we have, for all $s \in S_L$, that $(\chi(X) \setminus \{\xi\})_s \neq \{\bot\} \Rightarrow (\chi(Y) \setminus \{\xi\})_s \neq \{\bot\}$ and also that $\chi(X)_s R \chi(Y)_s$. As R is reflexive, closed under non-empty countable unions, and right-closed under $\leq$, we conclude.

2. $\texttt{if}\, b\, \texttt{then}\, C_{\mathtt{tt}}\, \texttt{else}\, C_{\mathtt{ff}}$: The case where $X = \{\bot\}$ is straightforward, using Lemma 6. Otherwise suppose that $X R^{\times} Y$. As $X \neq \{\bot\}$ we have $Y \neq \{\bot\}$. As b is public, and by Lemma 8, for every $\zeta \neq \bot$ such that $\zeta_{/\xi}$ is a store projection, $(w \mapsto [\![b]\!]^w_{\zeta(w)})_{/\xi}$ only depends on $S(\zeta_{/\xi})_L$, and can only be a boolean $t \in \mathbb{B}$ or ξ, independent of w. Define the store projection set $X_{\mathtt{tt}}$ to be

$$\bigcup_{s \in S} \{\zeta \in X | \zeta = (- \cdot s) \wedge \forall w.\, [\![b]\!]^w_{ws} = \mathtt{tt}\}{\downarrow} \cup \{\bot\}$$

and define the store projection set X_ξ to be

$$\{\zeta \in X \mid \zeta_{/\xi} = w \mapsto \xi\}{\downarrow} \cup \{\bot\}$$

and define the store projection sets $X_{\mathtt{ff}}$, $Y_{\mathtt{tt}}$, $Y_{\mathtt{ff}}$, and Y_ξ similarly. We have that $X = X_{\mathtt{tt}} \cup X_{\mathtt{ff}} \cup X_\xi$ and that at least one of $X_{\mathtt{tt}}$, $X_{\mathtt{ff}}$, or X_ξ is not $\{\bot\}$, and similarly for Y. We also have that

$$\begin{aligned}[\![\texttt{if}\, b\, \texttt{then}\, C_{\mathtt{tt}}[c_0]\, \texttt{else}\, C_{\mathtt{ff}}[c_0]]\!]^{\dagger}(X) = & \\ & [\![C_{\mathtt{tt}}[c_0]]\!]^{\dagger}(X_{\mathtt{tt}}) \cup [\![C_{\mathtt{ff}}[c_0]]\!]^{\dagger}(X_{\mathtt{ff}}) \cup X_\xi \qquad (*)\end{aligned}$$

and similarly for Y.
Similarly to the previous point, we have:

$$\chi(X_{\mathtt{tt}})_{s'} = \bigcup_{s \in S} \{\chi(X)_{s'} \mid \forall w.\, [\![b]\!]^w_{ws} = \mathtt{tt} \text{ and } s_L = s'\}{\downarrow}$$

and can check that $X_{\mathtt{tt}} R^{\times} Y_{\mathtt{tt}}$, $X_{\mathtt{ff}} R^{\times} Y_{\mathtt{ff}}$, and $X_{\xi} R^{\times} Y_{\xi}$, making use of the facts that $X \leq_L Y$, and that R is reflexive, closed under non-empty countable unions, and right-closed under $\leq$. We can then conclude using $(*)$ and the fact that R is closed under binary unions.

3. $\mathtt{skip}$, $C'; C''$, $C' + C''$, $\mathtt{while}\, b\, \mathtt{do}\, C_w$, or $[\,]$: In all these cases the proof is analogous to that of the corresponding parts of the proof of Lemma 1.

This concludes the proof as $[\![C[c_0]]\!] \preceq [\![C[c_1]]\!]$ implies $[\![C[c_0]]\!] \leq_L [\![C[c_1]]\!]$.

We need some further lemmas before proving the second half of the coincidence.

Lemma 10. *Let c be a low-level command such that $[\![c]\!]$ preserves store projection. Let $C[\,]$ be a public command context. Then $[\![C[c]]\!]$ preserves store projections.*

Proof. The proof is an induction on public command contexts, and is similar to, but simpler than, the proof of Lemma 9.

Let $-^{\chi}$ be the map from relations on $\mathcal{H}(S_{\xi\perp})$ to relations on store projection sets left anonymous in the main text. That is, for R a relation on $\mathcal{H}(S_{\xi\perp})$:

$$X R^{\chi} Y \equiv_{\text{def}} \chi(X)\, R\, \chi(Y)$$

We define $\varpi : S_{\xi\perp} \to \mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$ by:

$$\begin{aligned} \varpi(\perp) &= \{w \mapsto \perp\}\downarrow \\ \varpi(s) &= \{- \cdot s\}\downarrow \\ \varpi(\xi) &= \{\zeta | P(\zeta(w) = \xi) \geq \delta\}\downarrow \end{aligned}$$

Note that, for every $X \in S_{\xi\perp}$, we have $\chi(\varpi(X)) = X$.

Lemma 11. *1. Let R_i $(i \geq 0)$ be relations on $\mathcal{H}(S_{H\xi\perp})$ such that if $X R_i Y$ holds, then $\xi \in X$ implies that $\xi \in Y$ and $(X \setminus \{\xi\}) \neq \{\perp\}$ implies that $(Y \setminus \{\xi\}) \neq \{\perp\}$). Let R be the closure of the union of the R_i under increasing ω-sups, binary union, and right-closure under $\leq$. Then R^+ is the closure of the union of the relations R_i^+ under increasing ω-sups, binary unions, and right-closure under $\leq$.*

2. *Let R_i $(i \geq 0)$ be relations on $\mathcal{H}(S_{\xi\perp})$ and let R be their closure under increasing ω-sups, binary unions, and right-closure under $\leq$. Then R^{χ} is the closure of the union of the R_i^{χ} under increasing ω-sups, binary unions, and right-closure under $\leq$ (restricted to store projection sets).*

Proof. 1. The proof is almost exactly the same as for Lemma 2. As $-^+$ is evidently monotone, R^+ contains the R_i^+. Next, as we know, if a relation S on $\mathcal{H}(S_{H\xi\perp})$ is closed under any one of increasing ω-sups, binary unions, or right-closure under $\leq$, then so is S^+. So we also have that R^+ is closed under increasing ω-sups and binary unions, and is right-closed under $\leq$. It is therefore included in the closure of the union of the R_i^+ under increasing ω-sups, binary unions, and right-closure under $\leq$.

For the converse, suppose that UR^+W to show that U and W are related in the closure of the union of the R_i^+ under increasing ω-sups, binary unions, and right-closure under $\leq$. For any given s in S_L, by definition of $-^+$, U_sRW_s, and so, by the definition of R, there is a set $J^{(s)} \subseteq \mathbb{N}$, and relations $X_j^{(s)} R_{i_j^{(s)}} Y_j^{(s)}$ such that $U_s = \bigcup_{j\in J^{(s)}} X_j^{(s)}$ and $W_s \supseteq \bigcup_{j\in J^{(s)}} Y^{(s)}$. We may assume without loss of generality that the $J^{(s)}$ are disjoint.

Let

$$\begin{aligned}
J &= \textstyle\bigcup_{s\in S_L} J^{(s)} \\
X_j &= \{s' \mid s'_H \in X_j^{(s)}, s'_L = s\} \cup \{\xi \mid \xi \in X_j^{(s)}\} \cup \{\perp\} && (j \in J^{(s)}) \\
Y_j &= \{s' \mid s'_H \in Y_j^{(s)}, s'_L = s\} \cup \{\xi \mid \xi \in Y_j^{(s)}\} \cup \{\perp\} && (j \in J^{(s)}) \\
i_j &= i_j^{(s)} && (j \in J^{(s)})
\end{aligned}$$

We verify that $(X_j)_s = X_j^{(s)}$, if $j \in J^{(s)}$, and $= \{\perp\}$, otherwise, and similarly for the $(Y_j)_s$. Consequently, $U = \bigcup_j X_j$, $W \supseteq \bigcup_j Y_j$, and for all s in S_L, $(X_j)_s R_{i_j} (Y_j)_s$. Since, by hypothesis, if XR_iY holds then $\xi \in X$ implies $\xi \in Y$ and $(X \setminus \{\xi\}) \neq \{\perp\}$ implies $(Y \setminus \{\xi\}) \neq \{\perp\}$, we note that $X_j R_{i_j}^+ Y_j$. We conclude that U and W are related as required.

2. As $-^\chi$ is evidently monotone, R^χ contains the R_i^χ. Next one can check that if a relation S on $\mathcal{H}(S_{\xi\perp})$ is closed under either one of increasing ω-sups or binary unions, then so is S^χ, and that if it is right-closed under $\leq$ then S^χ is right-closed under $\leq$ restricted to store projection sets; to do this one uses the fact that χ is monotone and preserves increasing ω-sups and binary unions. So we also have that R^χ is closed under increasing ω-sups and binary unions, and right-closed under $\leq$ restricted to store projection sets.

For the converse, suppose that $UR^\chi W$ to show that U and W are related in the closure of the union of the R_i^χ under increasing ω-sups, binary unions, and right-closure under $\leq$, restricted to store projection sets.

We have $\chi(U)\, R\, \chi(W)$. So, by the definition of R, for some nonempty $J \subseteq \mathbb{N}$, there are relations $X_j R_{i_j} Y_j$ such that both $\chi(U) = \bigcup_j X_j$ and $\chi(W) \supseteq \bigcup_j Y_j$ hold. One can show that, for $j \in J$, $\chi(U \cap \varpi(X_j)) = X_j$ and $\chi(W \cap \varpi(Y_j)) = Y_j$. So we have $(U \cap \varpi(X_j)) R^{\chi}_{i_j} (W \cap \varpi(Y_j))$, for $j \in J$. So, calculating that:

$$\bigcup_i (U \cap \varpi(X_j)) = U \cap \varpi(\bigcup_i X_j) = U \cap \varpi(\chi(U)) = U$$

and that:

$$\bigcup_i (W \cap \varpi(Y_j)) = W \cap \varpi(\bigcup_i Y_j) \subseteq W \cap \varpi(\chi(W)) = W$$

we see that U and W are related as required.

We can now establish the second half of the coincidence. We will use notations $[c_x \mid x \in \mathrm{PubLoc}]$ and $\mathtt{if}\, b\, \mathtt{then}\, c$ for low-level commands analogous to those used for high-level commands in the proof of Lemma 3.

Lemma 12. *Let c and c' be two commands of the low-level language such that $c \sqsubseteq_L c'$, and $[\![c]\!]$ and $[\![c']\!]$ preserve store projections. Then $[\![c]\!] \preceq [\![c']\!]$ holds.*

Proof. Let c_0 and c_1 be two commands such that $c_0 \sqsubseteq_L c_1$ and that $[\![c_0]\!]$ and $[\![c_1]\!]$ preserve store projections. (The latter property, in combination with Lemmas 10 and 7, allows us to assume that we are always dealing with store projection sets.) We define relations R_i on S_H (for $i \geq 0$) as follows:

- for every $X \in \mathcal{H}(S_{H\perp})$, we have $X\ R_0\ X$;
- for every $X, Y \in \mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$ such that $X\ R_i^{\times}\ Y$, we have $(\chi([\![c_0]\!]^{\dagger}(X)))_s\ R_{i+1}\ (\chi([\![c_1]\!]^{\dagger}(Y)))_s$, forall $s \in S_L$.

We first prove by induction on i that, if $X\ R_i^{\times}\ Y$, then, for every $s \in S$ such that $\zeta = -\cdot s \in X$, there exists a public command context C and an $s_0 \in S$ such that $\zeta \in ([\![C[c_0]]\!](-\cdot s_0))$ and $\chi([\![C[c_1]]\!](-\cdot s_0))_{s_L} \subseteq \chi(Y)_{s_L}$.

- Suppose $X\ R_0^{\times}\ Y$. Take $C = \mathsf{skip}$ and $s_0 = s$ (hence $-\cdot s_0 = \zeta$). Then $\zeta \in [\![\mathsf{skip}]\!](\zeta)$ and $\chi([\![\mathsf{skip}]\!](\zeta)) = \{s, \perp\} \subseteq \chi(X) = \chi(Y)$, and we conclude by monotonicity of $-_{s_L}$.
- Suppose that $X\ R_{i+1}^{\times}\ Y$. By definition of $X\ R_{i+1}^{\times}\ Y$, and in particular $\chi(X)_{s_L}\ R_{i+1}\ \chi(Y)_{s_L}$, there exist $X'\ R_i^{\times}\ Y'$ and $s' \in S_L$ such that, $\chi([\![c_0]\!]^{\dagger}(X'))_{s'} = \chi(X)_{s_L}$ and $\chi([\![c_1]\!]^{\dagger}(Y'))_{s'} = \chi(Y)_{s_L}$.

We have that $\chi([\![c_0]\!]^\dagger(X')) = S(\{\zeta''_{/\xi} \mid \zeta'' \in \max([\![c_0]\!]^\dagger(X'))\})\downarrow$ and also $s_H \in \chi(X)_{s_L}$. Hence, using the definition of $-^\dagger$, there exist a $\zeta' \in \max(X')$ and a $\zeta'' \in \max([\![c_0]\!](\zeta'))$ such that $S(\zeta'')_L = s'$ and $S(\zeta'')_H = s_H$. By Lemma 6, as $\zeta'' \neq \bot$ we have $\zeta' \neq \bot$, and as $P(\zeta''(w) = \xi) = 0$, we have $P(\zeta'(w) = \xi) = 0$.
Applying the induction hypothesis to $X'\ R_i^\times\ Y'$ and $S(\zeta')$, there is a public command context C and an $s_0 \in S$ s.t. $\zeta' \in [\![C[c_0]]\!](-\cdot s_0)$ and $\chi([\![C[c_1]]\!](-\cdot s_0))_{S(\zeta')_L} \subseteq \chi(Y')_{S(\zeta')_L}$.
We consider the public command context

$$\begin{aligned} C' =_{\text{def}} C;\ & \\ & [\texttt{if}\ !x_{\texttt{nat}} \neq S(\zeta')_L(x)\ \texttt{then}\ \Omega \mid x \in \text{PubLoc}]; \\ & [\,];[x := S(\zeta)_L(x) \mid x \in \text{PubLoc}] \end{aligned}$$

We have $\zeta' \in [\![C[c_0]]\!](-\cdot s_0)$, so ζ'' is in

$$[\![C[c_0]; [\texttt{if}\ !x_{\texttt{nat}} \neq S(\zeta')_L(x)\ \texttt{then}\ \Omega \mid x \in \text{PubLoc}]; c_0]\!](-\cdot s_0)$$

so ζ is in $[\![C'[c_0]]\!](-\cdot s_0)$.
Also, $\chi([\![C[c_1]]\!](-\cdot s_0))_{S(\zeta')_L} \subseteq \chi(Y')_{S(\zeta')_L}$, hence

$$\begin{aligned} &\max([\![C[c_1]; [\texttt{if}\ !x_{\texttt{nat}} \neq S(\zeta')_L(x)\ \texttt{then}\ \Omega \mid x \in \text{PubLoc}]]\!](-\cdot s_0))_{/\xi} \\ &\quad \subseteq \max(Y')_{/\xi} \end{aligned}$$

hence,

$$\begin{aligned} &\chi([\![C[c_1]; [\texttt{if}\ !x_{\texttt{nat}} \neq S(\zeta')_L(x)\ \texttt{then}\ \Omega \mid x \in \text{PubLoc}]; c_1]\!](-\cdot s_0))_{s'} \\ &\quad \subseteq\ \chi([\![c_1]\!]^\dagger(Y'))_{s'} = \chi(Y)_{s_L} \end{aligned}$$

and hence (we replace the low variables by the corresponding values in s_L)

$$\chi([\![C'[c_1]]\!](-\cdot s_0))_{s_L} \subseteq \chi(Y)_{s_L}$$

We now prove that:

$$X R_i^\times Y\ \Rightarrow\ \xi \in \chi([\![c_0]\!]^\dagger(X)) \Rightarrow \xi \in \chi([\![c_1]\!]^\dagger(Y))$$

and that

$$\begin{aligned} X R_i^\times Y \Rightarrow\ & \forall s \in S_L.\, ((\chi([\![c_0]\!]^\dagger(X)))_s \setminus \{\xi\}) \neq \{\bot\} \\ & \Rightarrow ((\chi([\![c_1]\!]^\dagger(Y)))_s \setminus \{\xi\}) \neq \{\bot\} \end{aligned}$$

The two proofs are similar, first, for ξ. If $\xi \in \chi([\![c_0]\!]^\dagger(X))$, by definition of $-^\dagger$, there exists $\zeta \in \max(X)$ such that $\xi \in \chi([\![c_0]\!]^\dagger(\zeta))$. In the case that

$P(\zeta(w) = \xi) \geq \delta$, then the proof is by Lemma 6; otherwise $\zeta = - \cdot s$, for some $s \in S$ (it cannot be $\bot$). We know from the above that, since $X R_i^{\times} Y$, there exists a public command context C and an $s_0 \in S$ such that $\zeta \in [\![C[c_0]]\!](- \cdot s_0)$ and $\chi([\![C[c_1]]\!](- \cdot s_0))_{s_L} \subseteq \chi(Y)_{s_L}$. We let

$$C' =_{\text{def}} C; [\texttt{if}\ !x_{\texttt{nat}} \neq S(\zeta)_L(x)\ \texttt{then}\ \Omega \mid x \in \text{PubLoc}]$$

As $\chi([\![C[c_1]]\!](- \cdot s_0))_{s_L} \subseteq \chi(Y)_{s_L}$, $\chi([\![C'[c_1]]\!](- \cdot s_0)) \subseteq \chi(Y)$, and so $\chi([\![C'[c_1]; c_1]\!](- \cdot s_0) \subseteq \chi([\![c_1]\!]^{\dagger}(Y))$.

We have $\xi \in \chi([\![C'[c_0]; c_0]\!](- \cdot s_0))$. Also, as $C'; [\ ]$ is a public command context, $[\![C'[c_0]; c_0]\!](- \cdot s_0) \leq_L [\![C'[c_1]; c_1]\!](- \cdot s_0)$. Hence we have $\xi \in \chi([\![C'[c_1]; c_1]\!](- \cdot s_0))$. This concludes the proof, as we have $\chi([\![C'[c_1]; c_1]\!](- \cdot s_0)) \subseteq \chi([\![c_1]\!]^{\dagger}(Y))$.

Now, for $\bot$. Suppose $\chi([\![c_0]\!]^{\dagger}(X))_s \setminus \{\xi\} \neq \{\bot\}$ for some $s \in S_L$. Then there exists $s' \in S$ such that $s'_L = s_L$ and $- \cdot s' \in [\![c_0]\!]^{\dagger}(X)$. By definition of $-^{\dagger}$, there exists $\zeta \in \max(X)$ such that $- \cdot s' \in [\![c_0]\!]^{\dagger}(\zeta)$. If ζ is not a store projection, then there is a contradiction by Lemma 6, otherwise, $\zeta = - \cdot s''$, for some $s'' \in S$.

We know from the above that, since $X R_i^{\times} Y$, there exists a public command context C and an $s_0 \in S$ such that both $\zeta \in [\![C[c_0]]\!](- \cdot s_0)$ and $\chi([\![C[c_1]]\!](- \cdot s_0))_{s''_L} \subseteq \chi(Y)_{s''_L}$ hold. We let

$$C' =_{\text{def}} C; [\texttt{if}\ !x_{\texttt{nat}} \neq S(\zeta)_L(x)\ \texttt{then}\ \Omega \mid x \in \text{PubLoc}]$$

As $\chi([\![C[c_1]]\!](- \cdot s_0))_{s''_L} \subseteq \chi(Y)_{s''_L}$, $\chi([\![C'[c_1]]\!](- \cdot s_0)) \subseteq \chi(Y)$ holds, and so too, therefore, does $\chi([\![C'[c_1]; c_1]\!](- \cdot s_0) \subseteq \chi([\![c_1]\!]^{\dagger}(Y))$.

We have $- \cdot s' \in \chi([\![C'[c_0]; c_0]\!](- \cdot s_0))$. Also, since $C'; [\ ]$ is a public command context, we have

$$[\![C'[c_0]; c_0]\!](- \cdot s_0) \leq_L [\![C'[c_1]; c_1]\!](- \cdot s_0)$$

Hence we have $- \cdot s''' \in \chi([\![C'[c_1]; c_1]\!](- \cdot s_0))$ for some s''' with $s'''_L = s'_L = s$. As $\chi([\![C'[c_1]; c_1]\!](- \cdot s_0)) \subseteq \chi([\![c_1]\!]^{\dagger}(Y))$, this concludes the proof.

By the definition of R_{i+1}, we have

$$X\ R_i^{\times}\ Y \Rightarrow \forall s \in S_L.\, (\chi([\![c_0]\!]^{\dagger}(X)))_s R_{i+1} (\chi([\![c_1]\!]^{\dagger}(Y)))_s$$

From both facts and the definition of R_{i+1} we then deduce that

$$\forall X, Y \in \mathcal{H}(\text{Mem}^W_{\xi\bot}).\, X R_i^{\times} Y \Rightarrow [\![c_0]\!]^{\dagger}(X) R_{i+1}^{\times} [\![c_1]\!]^{\dagger}(Y)$$

We now define R as the closure under increasing ω-sups, right-closure under $\leq$ and closure under binary unions of the union of the R_i. We then conclude, using Lemma 11, that $[\![c_0]\!] \preceq_R [\![c_1]\!]$.

Combining Lemmas 9 and 12, we obtain the desired coincidence:

Theorem 4. *Let c and c' be two commands of the low-level language such that $[\![c]\!]$ and $[\![c']\!]$ preserve store projections. Then $c \sqsubseteq_L c'$ holds if and only if $[\![c]\!] \preceq [\![c']\!]$ does.*

Example 6. Suppose that there is only one private location, and consider the two commands:

$$c_4 = (1{:=}1) \ + \ (2{:=}1) \ + \ (3{:=}1) \ + \ (4{:=}1) \qquad c_6 = (1{:=}1); (2{:=}1)$$

As seen above, we have that $[\![c_4]\!](\zeta_i) = \{\zeta^1_\xi, \zeta^2_\xi, \zeta^3_\xi, \zeta^4_\xi\}\downarrow$. We also have that $[\![c_6]\!](\zeta_i) = \{w \mapsto \xi\}\downarrow$. Since $P(\zeta^i_\xi(w) = \xi) \geq \delta$, we can verify that c_4 and c_6 are equivalent. (Thus, a nondeterministic guess is no better than failure.) □

5 High and Low

In this section we investigate the relation between the high-level language and the low-level language. Specifically, we define a simple translation from the high-level language to the low-level language, then we study its properties.

We define the compilation of high-level commands c (expressions e, boolean expressions b) to low-level commands $c^\downarrow$ (expressions $e^\downarrow$ and boolean expressions $b^\downarrow$) by setting:

$$\begin{aligned} (!l_{\mathtt{loc}})^\downarrow &= !l_{\mathtt{nat}} \\ (l_{\mathtt{loc}} := e)^\downarrow &= l_{\mathtt{nat}} := e^\downarrow \end{aligned}$$

and proceeding homomorphically in all other cases, for example setting:

$$(e + e')^\downarrow = e^\downarrow + e'^\downarrow$$

Crucially, this compilation function, which is otherwise trivial, transforms high-level memory access to low-level memory access.

We begin with two lemmas about compilation.

Lemma 13. *1. Let e be a high-level natural-number expression. Then, for every $s \in S$, and $w \in W$,*

$$[\![e^\downarrow]\!]^w_{w \cdot s} = [\![e]\!](s)$$

2. Let b be a high-level boolean expression. Then, for every $s \in S$, and $w \in W$,

$$[\![b^\downarrow]\!]^w_{w \cdot s} = [\![b]\!](s)$$

3. Let c be a high-level command. Then, for every $s \in S$,

$$[\![c^{\downarrow}]\!](-\cdot s) = \{-\cdot s' \mid s' \in [\![c]\!](s)\}\downarrow$$

Proof. The first two parts are straightforward structural inductions on natural number and boolean expressions. For the third we proceed by structural induction on commands:

1. $l_{\mathtt{loc}} := e$: The result is immediate by part 1 and the definition of the semantics.
2. $\mathtt{if}\, b\, \mathtt{then}\, c_{\mathrm{tt}}\, \mathtt{else}\, c_{\mathrm{ff}}$: By part 2, we have $[\![b]\!](s) = [\![b^{\downarrow}]\!]^{w}_{w\cdot s} = t$ with $t \in \mathbb{B}$, hence
   - $[\![c]\!](s) = [\![c_t]\!](s)$, and
   - $[\![c^{\downarrow}]\!](-\cdot s) = [\![c_t^{\downarrow}]\!](-\cdot s)$

   The result then follows by applying the induction hypothesis to c_t.
3. skip: Here $\eta(-\cdot s) = \{-\cdot s\}\downarrow$ and $\eta(s) = \{s\}\downarrow$.
4. $c'; c''$: The result follows from the definition of $-^{\dagger}$ and applying the induction hypothesis to c' and c''.
5. $c' + c''$: The result follows by applying the induction hypothesis to c and c'.
6. $\mathtt{while}\, b\, \mathtt{do}\, c_w$: Define iterates $c^{(n)}$ of $\mathtt{while}\, b\, \mathtt{do}\, c_w$ by setting $c^{(0)} = \Omega$ and $c^{(n+1)} = \mathtt{if}\, b\, \mathtt{then}\, \mathsf{skip}\, \mathtt{else}\, c_w; c^{(n)}$, where Ω is some command denoting $\bot$, as does its compilation. Note that the $(c^{(n)})^{\downarrow}$ are the corresponding iterates of $(\mathtt{while}\, b\, \mathtt{do}\, c_w^{\downarrow})$.
   By induction on n, we have

   $$[\![(c^{(n)})^{\downarrow}]\!](-\cdot s) = \{-\cdot s' \mid s' \in [\![c^{(n)}]\!](s)\}\downarrow$$

   as the case $n = 0$ follows from the fact that Ω and its compilation denote (the relevant) $\bot$, and the induction step follows using the same reasoning as in the second, third, and fourth cases.
   The conclusion follows, as we have

   $$[\![c]\!] = \bigvee_{n\geq 0} [\![c^{(n)}]\!] \quad \text{and} \quad [\![c^{\downarrow}]\!] = \bigvee_{n\geq 0} [\![(c^{(n)})^{\downarrow}]\!]$$

   the latter by the above remark on the iterates of $(\mathtt{while}\, b\, \mathtt{do}\, c_w)^{\downarrow}$.

This concludes the proof.

Lemma 14. *Let c be a high-level command. Then $[\![c^{\downarrow}]\!]$ preserves store projections, and for every store projection set X we have:*

$$\chi([\![c^{\downarrow}]\!]^{\dagger}(X)) = [\![c]\!]^{\dagger}(\chi(X) \setminus \{\xi\}) \cup \{\xi \mid \xi \in \chi(X)\}$$

Proof. This lemma is a straightforward consequence of Lemmas 6 and 13.

Theorem 5 relates the simulation relations of the two languages. It states that a high-level command c simulates another high-level command c, with respect to all public contexts of the high-level language, if and only if the compilation of c simulates the compilation of c', with respect to all public contexts of the low-level language.

Theorem 5. *Let c and c' be two high-level commands. Then $[\![c]\!] \preceq [\![c']\!]$ holds if and only if $[\![c^{\downarrow}]\!] \preceq [\![c'^{\downarrow}]\!]$ does.*

Proof. In one direction, let c and c' be commands such that $[\![c]\!] \preceq_{R_0} [\![c']\!]$, with R_0 a reflexive relation closed under increasing ω-sups, right-closed under $\leq$, and closed under binary unions. Let R be the closure of R_0 in $\mathcal{H}(S_{H\xi\perp})$ by reflexivity, increasing ω-sups, binary union, and right-closure under $\leq$. That is, XRY holds if both $(X \setminus \{\xi\})\, R_0\, (Y \setminus \{\xi\})$ and $\xi \in X \Rightarrow \xi \in Y$ do. Note that XR^+Y if $(X \setminus \{\xi\})\, R_0^+\, (Y \setminus \{\xi\})$ and $\xi \in X \Rightarrow \xi \in Y$. Let X and Y in $\mathcal{H}(\mathrm{Mem}^W_{\xi\perp})$ be such that $XR^{\times}Y$. We have to show that $[\![c^{\downarrow}]\!]^{\dagger}(X) R^{\times} [\![c'^{\downarrow}]\!]^{\dagger}(Y)$. By Lemma 14, $[\![c^{\downarrow}]\!]^{\dagger}(X)$ and $[\![c'^{\downarrow}]\!]^{\dagger}(Y)$ are store projection sets, and so this is equivalent to showing that

$$\chi([\![c^{\downarrow}]\!]^{\dagger}(X))\, R^+\, \chi([\![c'^{\downarrow}]\!]^{\dagger}(Y))$$

Using Lemma 14 again, we see that this latter statement is equivalent to:

$$([\![c]\!]^{\dagger}(\chi(X) \setminus \{\xi\}) \cup \{\xi \mid \xi \in \chi(X)\})\, R^+\, ([\![c']\!]^{\dagger}(\chi(Y) \setminus \{\xi\}) \cup \{\xi \mid \xi \in \chi(Y)\})$$

which in turn is equivalent, by a previous remark, to

$$[\![c]\!]^{\dagger}(\chi(X) \setminus \{\xi\})\, R_0^+\, [\![c']\!]^{\dagger}(\chi(Y) \setminus \{\xi\}) \;\wedge\; (\xi \in \chi(X) \Rightarrow \xi \in \chi(Y))$$

As $XR^{\times}Y$, we have that $\chi(X)\, R^+\, \chi(Y)$. It follows first that we have that $(\chi(X) \setminus \{\xi\})\, R_0^+\, (\chi(Y) \setminus \{\xi\})$, and then $[\![c]\!]^{\dagger}(\chi(X) \setminus \{\xi\})\, R_0^+\, [\![c']\!]^{\dagger}(\chi(Y) \setminus \{\xi\})$ (as $[\![c]\!] \preceq_{R_0} [\![c']\!]$); and second we have that $\xi \in \chi(X) \Rightarrow \xi \in \chi(Y)$. The conclusion follows.

In the other direction, let c and c' be two commands such that $[\![c^{\downarrow}]\!] \preceq_R [\![c'^{\downarrow}]\!]$, with R_0 a reflexive relation closed under increasing ω-sups, right-closed under $\leq$, and closed under binary unions. We let R be the restriction of R_0 to $\mathcal{H}(S_{H\perp})$. That is, XRY if XR_0Y. Note that XR^+Y if XR_0^+Y. Let X and Y in $\mathcal{H}(S_{\perp})$ be such that XR^+Y. Hence XR_0^+Y. We have $\varpi(X) R_0^{\times} \varpi(Y)$, hence, by the definition of $\preceq_{R_0}$, we have $[\![c^{\downarrow}]\!]^{\dagger}(\varpi(X)) R_0^{\times} [\![c'^{\downarrow}]\!]^{\dagger}(\varpi(Y))$.

By Lemmas 6 and 13, $[\![c^{\downarrow}]\!]^{\dagger}(\varpi(X)) = \varpi([\![c]\!]^{\dagger}(X))$, and similarly for Y. Thus, by the definition of $R_0^{\times}$, $[\![c]\!]^{\dagger}(X) R_0^+ [\![c']\!]^{\dagger}(Y)$, hence $[\![c]\!]^{\dagger}(X) R^+ [\![c']\!]^{\dagger}(Y)$, and we conclude.

Our main theorem, Theorem 6, follows from Theorem 5, the two previous theorems, and Lemma 14. Theorem 6 is analogous to Theorem 5, but refers to the contextual pre-orders: a high-level command c implements another high-level command c', with respect to all public contexts of the high-level language, if and only if the compilation of c implements the compilation of c', with respect to all public contexts of the low-level language.

Theorem 6 (Main theorem). *Let c and c' be two high-level commands. Then $c \sqsubseteq_L c'$ holds if and only if $c^{\downarrow} \sqsubseteq_L c'^{\downarrow}$ does.*

Theorem 6 follows from Theorem 5, the two previous theorems, and the lemma. The low-level statement is defined in terms of the probability δ that depends on the distribution on memory layouts. When δ is close to 1, the statement indicates that, from the point of view of a public context (that is, an attacker), the compilation of c behaves like an implementation of the compilation of c'. This implementation relation holds despite the fact that the public context may access memory via natural-number addresses, and thereby (with some probability) read or write private data of the commands. The public context may behave adaptively, with memory access patterns chosen dynamically, for instance attempting to exploit correlations in the distribution of memory layouts. The public context may also give "unexpected" values to memory addresses, as in practical attacks; the theorem implies that such behavior is no worse at the low level than at the high level.

For example, for the commands c_0 and c_1 of Example 1, the theorem enables us to compare how their respective compilations behave, in an arbitrary public low-level context. Assuming that δ is close to 1, the theorem basically implies that a low-level attacker that may access memory via natural-number addresses cannot distinguish those compilations. Fundamentally, this property holds simply because the attacker can read or write the location h considered in the example only with low probability.

6 Conclusion

A few recent papers investigate the formal properties of layout randomization, like ours [6–9]. They do not consider nondeterministic choice, and tend to reason operationally. However, the work of Jagadeesan et al. includes some semantic elements that partly encouraged our research; specifically, that work employs trace equivalence as a proof technique for contextual equivalence.

In this paper we develop a semantic approach to the study of layout randomization. Our work concerns nondeterministic languages, for which this approach has proved valuable in reconciling probabilistic choice with nondeterministic choice. However, the approach is potentially more general. In particular, the study of concurrency with nondeterministic scheduling would be an attractive next step. Also, extending our work to higher-order computation presents an interesting challenge.

References

1. Goldwasser, S., Micali, S.: Probabilistic encryption. JCSS **28** (1984) 270–299
2. Forrest, S., Somayaji, A., Ackley, D.H.: Building diverse computer systems. In: 6th Workshop on Hot Topics in Operating Systems. (1997) 67–72
3. Druschel, P., Peterson, L.L.: High-performance cross-domain data transfer. Technical Report TR 92-11, Department of Computer Science, The University of Arizona (1992)
4. PaX Project: The PaX project (2004) `http://pax.grsecurity.net/`.
5. Erlingsson, Ú.: Low-level software security: Attacks and defenses. In: FOSAD IV Tutorial Lectures. Volume 4677 of LNCS., Springer (2007) 92–134
6. Pucella, R., Schneider, F.B.: Independence from obfuscation: A semantic framework for diversity. Journal of Computer Security **18**(5) (2010) 701–749
7. Abadi, M., Plotkin, G.D.: On protection by layout randomization. ACM Transactions on Information and System Security **15**(2) (2012) 8:1–8:29
8. Jagadeesan, R., Pitcher, C., Rathke, J., Riely, J.: Local memory via layout randomization. In: 24th IEEE Computer Security Foundations Symposium. (2011) 161–174
9. Abadi, M., Planul, J.: On layout randomization for arrays and functions. In: POST. Volume 7796 of LNCS., Springer (2013) 167–185
10. Lincoln, P., Mitchell, J., Mitchell, M., Scedrov, A.: A probabilistic poly-time framework for protocol analysis. In: Proceedings of the Fifth ACM Conference on Computer and Communications Security. (1998) 112–121
11. Mitchell, J.C., Ramanathan, A., Scedrov, A., Teague, V.: A probabilistic polynomial-time process calculus for the analysis of cryptographic protocols. TCS **353**(1-3) (2006) 118–164
12. Canetti, R., Cheung, L., Kaynar, D.K., Liskov, M., Lynch, N.A., Pereira, O., Segala, R.: Analyzing security protocols using time-bounded task-pioas. Discrete Event Dynamic Systems **18**(1) (2008) 111–159
13. Gierz, G., Hofmann, K.H., Keimel, K., Lawson, J.D., Mislove, M.W., Scott, D.S.: Continuous lattices and domains. Volume 93 of Encyclopaedia of mathematics and its applications. CUP (2003)
14. Mislove, M.W.: On combining probability and nondeterminism. ENTCS **162** (2006) 261–265
15. Tix, R., Keimel, K., Plotkin, G.D.: Semantic domains for combining probability and non-determinism. ENTCS **222** (2009) 3–99
16. Goubault-Larrecq, J.: Prevision domains and convex powercones. In: FoSSaCS. Volume 4962 of LNCS., Springer (2008) 318–333
17. Abadi, M., Planul, J., Plotkin, G.: Layout randomization and nondeterminism. ENTCS **298** (2013) 29–50

18. Abadi, M., Lamport, L.: The existence of refinement mappings. TCS **82**(2) (1991) 253–284
19. Sabelfeld, A., Sands, D.: Probabilistic noninterference for multi-threaded programs. In: 13th IEEE Computer Security Foundations Workshop. (2000) 200–214
20. Klarlund, N., Schneider, F.B.: Proving nondeterministically specified safety properties using progress measures. Information and Computation **107**(1) (1993) 151–170
21. de Roever, W.P., Engelhardt, K.: Data Refinement: Model-oriented Proof Theories and their Comparison. Volume 46 of Cambridge Tracts in Theo. Comp. Sci. CUP (1998)
22. Jackson, M.: A sheaf theoretic approach to measure theory. PhD thesis, U. Pitt. (2006)